Paper 189-25

WEB INTERFACING: MERGING STATISTICAL PROCESSING WITH INTERNET BASED READ AND WRITE PROCEDURES

William J. Montelpare, Ph.D., and Moira N. McPherson, Ph.D., School of Kinesiology, Lakehead University, Thunder Bay, Ontario, Canada

ABSTRACT

The purpose of this project was to develop a system to enter, store, retrieve, analyze and report data entirely over the Internet. The selected approach merged the Netscape Enterprise® Server's capability to create "applications" using client-side and server-side JavaScript with the statistical processing features of SAS® through the web-based software: SAS IntrNet™.

The application was designed to collect measures of physical health for a cohort of undergraduate students. The measures were first processed on "the client-side" to provide personal feedback. A request link enabled the client to send their measures to the "server" using "server-side" JavaScript write commands. The data were appended to a storage data set comprised of responses from visitors that agreed to submit their data.

SAS programs were written to provide descriptive statistics and tests of inference for the data set. A menu of available SAS programs was posted on the homepage web-site. Selecting a specific hot link, users could invoke a particular SAS program to analyze the data set. Each SAS program produced an output file that was posted to the Internet for the user to review. The results could then be compared to normative values available through the homepage menu of the web-site.

INTRODUCTION

The Internet, which includes the World Wide Web, as well as several other "specialized networks", is comprised of multiple forms of electronic media servers connecting databases, list servers, and task-specific processors. The Internet continues to be the most popular information delivery network because of its versatility and virtual structure. The amount and growth of available information through the Internet is staggering, and as individuals become more involved with the uses of the Internet as a tool for research and presentation, they quickly realize the volume of data that are so easily accessible. Yet the Internet has evolved beyond its role as a medium to disseminate information. Web authors can now incorporate the data processing feature of SAS IntrNet into their web site to extend the Internet's capabilities as a pathway for data processing.

STAGE 1: EXTENDING THE INTERNET BEYOND A REPOSITORY FUNCTION

Through HyperText Markup Language (HTML), individuals can post facts or information to a "web site". In turn, these "bits of information" can be retrieved by passive "browsers". Under this scenario, a "web author" can provide general information to all potential "visitors", or specific information to a designated group of individuals. In health research and the delivery of health services, providing essential information to a specific group of users has tremendous implications toward reducing costs while distributing timely, critical information. Using the Internet, the web author can ensure that necessary documentation is distributed to pre-designated individuals, simultaneously, without the restrictions of travel and postage.

Figure 1. The basic structure of a web page

```
<html>
<head> <title>
      title across the top of the browser
          </title>
</head>
  <body>
     My first message to the World.
    <P>
      Hello World!
    <P>
  </body>
</html>
```

Yet, the posting of documentation to a web-site is merely an electronic billboard approach to the dissemination of information. Web authors can however add dynamic characteristics to the web-site by incorporating the scripting features of an interpreted language, or the more complex interface features of compiled programs.

CLIENT-SIDE PROGRAMMING AND THE DEVELOPMENT OF "WEBULATORS"

Scripts are computer commands or statements which are "interpreted" by computers. Scripts are passed directly to the user (also referred to as the "client"), along with the HTML document when the client accesses the web page during an Internet session. JavaScript is one type of interpreted scripting language. JavaScript "commands" are not compiled but are embedded in HTML document files and interpreted by the computer that browses a web-site (Montelpare and McPherson, 1999). Scripts, which are more complex computer language statements than HTML, can invoke actions by the browsing client. The embedded scripts are interpreted by the client's browser and processed accordingly.

Using the logic of "subroutine programming", web authors can create specific scripted data processing routines. The data processing commands provide functionality to the client's Internet session as illustrated in Figure 2 (Gesing and Schneider, 1997; Kent and Kent, 1996; Purcell and Mara, 1997).

Figure 2. Sample JavaScript commands to add functionality to a web page

```
<SCRIPT LANGUAGE="JAVASCRIPT">
// initialize the variables used to collect data in the form
          function compute(form)
          {
    if (form.cellA.value==null||form.cellA.value.length==0)
                    {form.cellA.value=0;}
    // create the new variable for the converted number
          var nGetcellA=(form.cellA.value*1);
          var wtkg=(nGetcellA/(2.2));
    // create the output of script function printed on web page
                {
                form.outkg.value=wtkg;
                }
          return;
          }
    // add a function to reset all variables in the form to empty
          function reset(form)
          {
```

```
            form.cellA.value="";
            form.outkg.value="";
        }
    </SCRIPT>
... HTML commands go here.
```

For example, as part of a presentation on the evaluation of "aerobic fitness", undergraduate students were organized into laboratory sessions in which they completed several field tests and a recommended "gold standard" laboratory test. Using a non-scripted web page the authors simply used HTML statements to describe the methodology of each field test, and the computational procedures to determine "aerobic capacity". However, by adding script commands, the web page authors provided a series of "web based calculators—the Webulators". A listing of the JavaScript and HTML to create the aerobic fitness *Webulator* for a one-mile walk test are available from the authors.

Figure 3. A webulator to compute aerobic fitness for a one-mile walk test

Things You Measure	Values observed
Your weight in lbs. =	
Your age in years =	
The time to walk one mile (minutes) =	
Your heart rate after walking (beats/min) =	

Click to select male or female

Male ⬭

Female ⬭

Oxygen Consumption (ml/kg·min^{-1}) = ☐

DATA CAPTURE ACROSS THE "WEB".

There are several ways to collect data across the Internet. One very common approach is through the use of the "common gateway interface" (CGI) commands that link "clients to servers". Examples of CGI programs include: PERL, TCL, and C++. Although CGI commands provide functionality to the delivery of web pages, CGI commands are cryptic (Montelpare and McPherson, 1999). An alternative to the CGI command sequences for web authors is the use of server-side scripting. Server-side scripting refers to compiled JavaScript commands. The compiled server-side JavaScript commands are embedded into HTML code and presented to the client via a posted web-page application. Compiled server-side JavaScript creates a route by which an individual can pass information to the web page author for storage within a particular data set.

Netscape Communications Inc. has provided a server-side JavaScript compiler in its "SuiteSpot Server" and "Fast-Track" server packages. Previously the compiler was a standalone product under the name: LiveWire™ . The server-side JavaScript compiler is an application building product that bypasses the traditional CGI. Further, because the server-side scripts are merely extensions of client-side scripts, the learning curve to implement the data capture routines using compiled JavaScript is considerably reduced. Using a simple standard such as compiled JavaScript enables the researchers to focus more on "**the data collected**" than on "**the collection of data**". Server-side scripting routines were essential to the present study to create the pathway by which students could compute specific measures of aerobic fitness and submit their data across the Internet (Figure 4) (Montelpare and McPherson, 1999).

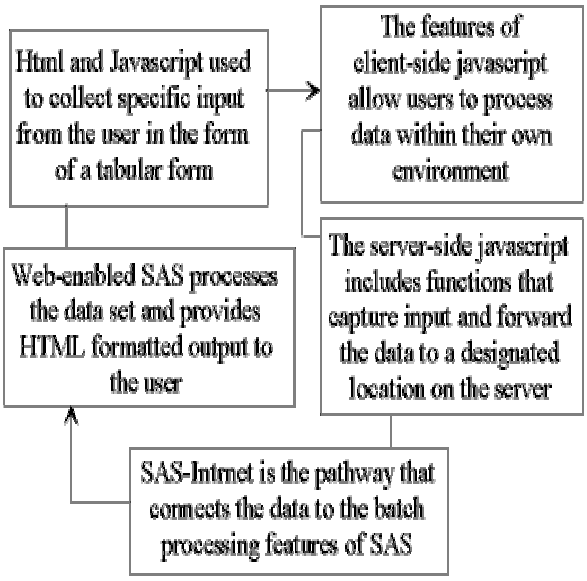
Figure 4. The data capture commands of Server-side JavaScript

```
<server>client.sex = request.gender; </server>
<server>client.oldname = request.newname; </server>
<server>client.stNum = request.studNum; </server>
<server>client.vo2k = request.type1VO2; </server>
<server>client.vo2d = request.type2VO2; </server>
<server>
newFile = new File("/instructors directory/outfile.dat")
newFile.open("a")
newFile.write(client.oldname + ", Student #=");
newFile.write(client.stNum + ", (sex)=");
newFile.write(client.sex + ", (VO2 Kline)=");
newFile.write(client.vo2k + ", (VO2 Dolgener)=");
newFile.writeln(client.vo2d);
newFile.close()
</server>
```

USING SAS FOR ONLINE ANALYTICAL PROCESSING: "OLAP".

In the beginning, there was batch programming. However, with the release of SAS IntrNet software researchers can not only analyze their data via web-based processing, but deliver the SAS output back to the "world" via the "web_out" features of the SAS InterNet software. The critical feature of SAS IntrNet is that anything you typically did in a "batch processing mode", you can do with SAS IntrNet. Figure 5 illustrates the flow of information from "client-side" interaction, through "server-side" data collection and server-side data storage, to SAS processing in batch mode.

Figure 5. A flow chart of information processing from client to Batch SAS output



In the initial stages of the present application, although SAS was used to analyze data that were submitted by the users, the process occurred in batch mode. That is, although client-side web pages were used to compute the aerobic fitness capacity (under different established regression equations), and "compiled JavaScript" was used to capture and store the data across the Internet. SAS was run in a batch mode using the "infile procedure" to read the data set on the server and produce various output (i.e. descriptive statistics, comparisons

of results across cohorts, and selected tests of inference). The process was mechanistically simple, but required an operator to process the data (run the batch program), retrieve the output and write the HTML to produce the report for the "client" to observe.

Online Analytical Processing--OLAP is the term used to describe the interactive analysis of data sets. SAS IntrNet® is a product from SAS Inc.which enables clients to use OLAP for their specific data sets. The concept of OLAP is simple. The web author produces all of the "proc statements" and SAS programs at the start of the project. An "infile statement" is included in each program to read the server-side storage data set(s). The SAS programs are stored in a SAS library accessible to SAS IntrNet, and the programs are hot linked to the authors web page.

Selecting a hot link from the main menu, the client can then run SAS programs for the analysis of the specific data set that they have appended information to. This allows the client to review SAS output from analyses of data sets that they have helped to produce during their Internet session. This process not only eliminates the need for an intermediate operator to run SAS and retrieve and present the output, but it also increases the speed of feedback to the clients.

HTML TO INVOKE SAS PROCESSING: SAS INTRNET

SAS Inc.® has contributed greatly to the use of the Internet as a tool for researchers. SAS Web Tools® provide researchers with the capability to post SAS listings or similar output to the Web for client browsing. While SAS IntrNet provides web clients with a powerful tool to analyze data sets instantaneously by running SAS programs over the web. The procedures and underlying theory of events in a SAS IntrNet session are well documented in the SAS IntrNet CD and on the SAS Inc. home page. Figure 6 presents a simple SAS IntrNet program to report information from data collected through a Web-based form to a server-side storage file.

Figure 3. A webulator to compute aerobic fitness for a one-mile walk test

```
options pagesize=55 linesize=80 center date;
*:
* begin capturing the output to be converted to HTML.
*;
%out2htm(capture=on,
        window=output,
        runmode=s);
data walk;
     infile '/export/home/WWW/docs/walk.dat' missover ;
    input  @1 name $ id gender $  kline dolgen;
if id = "undefined" then delete;
proc sort data=walk;by gender;
title 'Output from the heart health walking test';
proc print; var id gender kline dolgen;
proc chart; hbar kline dolgen;
run;
*;
* Stop the capturing and formatting of SAS output as HTML.
* The capturing procedure uses the default settings of SAS.
*;

data _null_;
  file _webout;
  put 'Content-type: text/html';
  put ;
  put '<HTML>';
  put '<HEAD><TITLE>Web interfacing</TITLE></HEAD>';
  put '<BODY bgcolor="white">';
  put '<center><br><Table cellpadding="0" border="0"
cellspacing="1">';
  put '<tr><td width=100% valign=middle align=center>';
  put '<a href="http://server/index.html"><font size=5><font
color=red>';
  put 'Click here to return to the home page</font> </a>';
  put '</td></tr></table></center><br>';
  put '</BODY>';
  put '</HTML>';
run;

*;
* Stop the capturing and formatting of the SAS output as
HTML.
* The capturing procedure uses the default settings of SAS.
*;

%out2htm(htmlfref=_WEBOUT,
        capture=off,
         window=output,
          openmode=replace,
                runmode=s);
```

CONCLUSION

It is important to recognize that the techniques presented here use standard formats and procedures which are easily adaptable in the creation and implementation of a "data application" web site for health researchers, health educators, and individuals delivering health services. Integrating JavaScript and SAS applications enables researchers, planners, and policy makers to build specific information management systems that can deliver feedback to clients. The feedback can be through client-side interpretation of results from computations and logic analysis of direct input, or through SAS analyses and web presentations of SAS output for the statistical analyses of larger data sets.

REFERENCES

Gesing, T., Schneider, J., (1997). *Javascript for the World Wide Web*, Berkeley: Peachpit Press.

Kent, P., Kent, J., (1996). *Official Netscape Javascript Book*, Research Triangle Park: Ventana Communications Group, Inc.

Montelpare, W.J., McPherson, M.N. (1999) Data Processing Across the Internet: A Model for Design. *The International Electronic Journal for Health Education*, Vol. 2(3), 127-137, July 1, 1999.

Purcell, L., Mara, M.J., (1997). *The ABCs of Javascript,* San Francisco: Sybex Inc.

CONTACT INFORMATION

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

William J. Montelpare, Ph.D., and Moira N. McPherson, Ph.D.,
School of Kinesiology, Lakehead University
955 Oliver Road
Thunder Bay, Ontario, P7B 5E1
(807) 343-8481 (807) 343-8640
Fax: (807) 343-8944
Email: wmontelp@flash.lakeheadu.ca,
Moira.McPherson@lakeheadu.ca.
Web: www.lakeheadu.ca/~kinesiology