Developing a PC-SAS® World Wide Web Database System

Faith Reneé Sloan
FRS Associates, San Francisco, California

ABSTRACT

The first few months of 1996 have seen an explosion of new Web-related technologies hitting the commercial marketplace as well as being offered freely at numerous sites on the Web. This is especially true in the world of low-cost Web servers, databases, and Web-database gateway software for desktop/personal computers. It is now quite possible to develop a functional Web site, complete with database applications, on a desktop computer using the SAS System in a matter of hours or a few days -- and often with only a small financial investment.

This paper presents and describes in detail such a system: Developing Educators Resource World Wide Web (WWW) Database System.

Background

The author was commissioned as a consultant to Hewlett-Packard to provide back-end programming to a site dedicated to developing educators. The system was developed using the Windows-NT 3.5 server as the operating system; Netscape Enterprise Server 2.0 as the web server; Informix as the database; and Netscape's LiveWire Pro as the javascript-based web application development tool.

The setup and installation wasn't as smooth as she would have liked and the time to production was lengthy indeed.

The author decided to embark upon an experiment to replicate a mini-version of this application using the SAS Software System.

The replicated system was designed, developed and tested in about eight hours! This was accomplished in spite of a lack of in-depth knowledge of all of the tools and components being employed. How was this possible? By taking advantage of increasingly sophisticated features within the SAS System, and by using the PERL programming language as the Common Gateway Interface.

Application Overview

The system described in this paper is an information system geared towards educators but can be applied to almost any query-type application. This project was undertaken to demonstrate the feasibility of designing and developing a low-cost, PC/SAS-based Web database while still providing a high level of functionality, capabilities, and growth potential.

The primary system requirements for the system were as follow:

1. **Use an existing personal computer**
2. **Use in-house software.**
3. **Use freely available tools and software to the extent possible.**
4. **Minimize the amount of programming required**
5. **Demonstrate the ability to obtain educators resources based on user-specified input.**
6. **Have fun!**

For these reasons the SAS Software System was chosen as the primary development tool. Also with the introduction of the experimental HTML formatting tools within SAS 6.11, the time from design to implementation was only about eight hours! The simplicity in installing the O'Reilly WebSite Pro server and the fact that it only costs a few hundred dollars, made it a hands-down winner in regards to web servers. The operating system of choice is the Windows NT 4.0 Server since it is the in-house platform.

Design

The Database:

Of course, the SAS Software was used to create the SAS database. The database contains two tables.

The Resource table contains all resources available to the educators. The fields are: the name of the resource; the type of resource (Books, WebSites, Models, etc.); business category code which is mapped using the Category table; synopsis, and the HTML file reference which facilitates access to the actual online resource. See Figure 1.

```
-----Alphabetic List of Variables and Attributes-----

    #  Variable  Type   Len   Pos   Label
    ------------------------------------------
    1  NAME      Char    60     0   Name
    2  TYPE      Char    15    60   Type
    3  CATEGCDE  Char     5    75   Category Code
    4  SYNOPSIS  Char   200    80   Synopsis
    5  HTMLFILE  Char   100   280   HTML Link
Sample Record:
Interactivity by Design Books DC How to Design whatever  <a
href='../sugi22/book5.html'>book5.html</a>
```

Figure 1: Resource Table Structure

The Category table is simply a lookup table for the business category codes (AA for

Analyze/Assess, DC for Design/Create, BB for Business Basics, etc.). See Figure 2.

HTML Page:

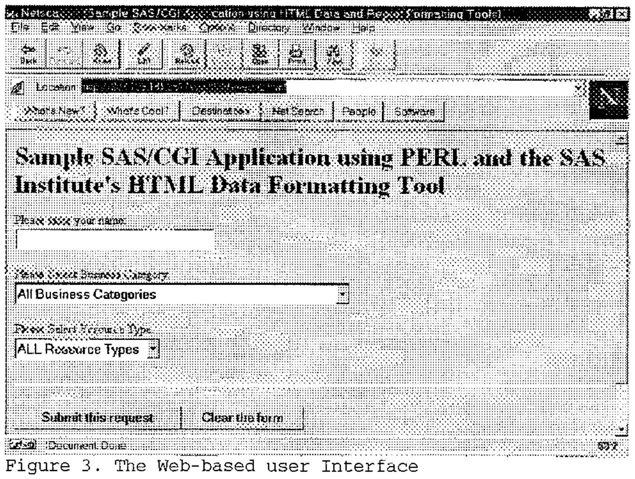
The HTML code to generate the main web page is as follow:

```
<HTML>
<HEAD>
   <TITLE>Sample SAS/CGI Application using HTML Data and
Report Formatting Tools</TITLE>
</HEAD>
<H1>Sample SAS/CGI Application using PERL and the SAS
Institute's HTML Data Formatting Tool</H1>
<P>
<FORM ACTION="http://207.105.181.251/cgi-shl/sugi22.pl"
METHOD=POST>
Please enter your name: <BR>
<INPUT type=TEXT size=30 name="username">
<P>
Please Select Business Category:<BR>
<SELECT name="categcde">
 <OPTION selected value="ALL">All Business Categories
 <OPTION value="AA">Analyze and Assess
 <OPTION value="ABR">Achieving Business Results
 <OPTION value="ALDP">Accelerating Learning, Development
and Performance
 <OPTION value="BB">Business Basics
 <OPTION value="C">consult
 <OPTION value="CF">Customer Focus
 <OPTION value="CG">Consulting
 <OPTION value="CM">Change Management
 <OPTION value="DC">Design/Create
 <OPTION value="EM">Evaluate/Measure
 <OPTION value="FI">Facilitate/Instruct
 <OPTION value="ID">Instructional Design
 <OPTION value="LDPT">Learning, Development, and
Performance Theory
 <OPTION value="ML">Manage/Lead
 <OPTION value="OB">Outsource/Broker
 <OPTION value="PMI">Project Management and
Implementation
 <OPTION value="PT">Performance Technology
 <OPTION value="SF">Scanning for the Future
</SELECT>
<p>
Please Select Resource Type:<BR>
<SELECT name="type">
 <OPTION selected value="ALL">ALL Resource Types
 <OPTION value="Books">Books
 <OPTION value="Models">Models
 <OPTION value="Presentation">Presentation
 <OPTION value="Research">Research
 <OPTION value="Skills">Skills
 <OPTION value="Templates">Templates
 <OPTION value="Training">Training
 <OPTION value="Vendors">Vendors
 <OPTION value="WebSites">WebSites
</SELECT>
<P>
<HR>
<P>
<INPUT TYPE=submit VALUE="Submit this request"><INPUT
TYPE="reset" VALUE="Clear the form"></FORM></P>
<BR CLEAR=ALL>
<HR SIZE=3 ALIGN=CENTER>
</BODY>
</HTML>
```

Using an HTML form for Database access and the application's user interface allowed us to use one text-based entry field for the user's name and two scrolled lists to select the business category and the type of resource one is interested in. The web page displayed to the user is located at http://207.105.181.251/sugi22/devedu.html and is illustrated in Figure 3.

```
-----Alphabetic List of Variables and Attributes-----

     #  Variable  Type   Len  Pos   Label
     ------------------------------------------
     1  CATEGDSC  Char    50   50   Description
     2  CATEGCDE  Char     5    0   Code
Sample Record:
Design/Create DC{PRIVATE }
```
Figure 2: Category Table Structure


Sample SAS/CGI Application using PERL and the SAS Institute's HTML Data Formatting Tool
File Edit View Go Bookmarks Options Directory Window Help
What's New? What's Cool? Destinations Net Search People Software
All Business Categories
ALL Resource Types
Submit this request
Clear the form
Document Done
Figure 3. The Web-based user Interface

When the user has made her selections, she can then click on the 'Submit this request button. This will send a 'request' to the Web server located at 207.105.181.251 to execute the SUGI22.PL PERL script (or program). This is referenced in the HTML FORM as <FORM ACTION="http://207.105.181.251/cgi-shl/sugi22.pl" METHOD=POST>. The three parameters which will be passed to SUGI22.PL to operate upon are name, categcde, and type.

Now we are ready to discuss the intricacies of the SUGI22.PL PERL program and the SAS Institute's HTML Data Formatting tool which is the essence of the Developing Educators Resource WWW Database System.

The PERL Script - SUGI22.PL:

Since this is not a PERL tutorial, this section will only focus on the code which actually creates the temporary pxxxx.html, pxxxx.sas, and pxxxx.log files.

The script below uses the process id and time to create the session's unique temporary id (xxxx). Being that there will potentially be many educators accessing the Web site at any one time, this ensures that a unique file will be 'served' to the each educator. Example file names are: p121852472586.sas, p121852472586.log, and p121852472586.html where $pname =p121852472586.

```
# Create unique name using process id and time
$pname="p$$".time;
# Output %LET statements to create SAS macro variables
# from the NAME=VALUE pairs.
open(OUTFI,">$PROGROOT/$pname.sas");
```

The pxxxx.sas file is opened in write mode in order to direct SAS source code to the file. Subsetting based upon the user selections for business category and resource type are performed before the d2htm macro is called. See the PERL script below.

The d2htm macro is a SAS Institute supplied macro supported under the SAS Software system version 6.11 and above. This is the HTML Data Formatting Tool! It will be discussed in more detail following this listing of SUGI22.PL.

```
#===========================================================
sub get_request {
    # Subroutine get_request reads the POST or GET form
    #request from STDIN into the variable  $request, and
    #then splits it        into its
    # name=value pairs in the associative array %rqpairs.
    # The number of bytes is given in the environment
    #variable
    # CONTENT_LENGTH which is automatically set by the
    #request generator.

    if ($ENV{'REQUEST_METHOD'} eq "POST") {
        read(STDIN, $request, $ENV{'CONTENT_LENGTH'});
    } elsif ($ENV{'REQUEST_METHOD'} eq "GET" ) {
        $request = $ENV{'QUERY_STRING'};
    }

    %rqpairs = ();
    @rqarray = &url_decode(split(/[&=]/, $request));
    while ( $key = shift(@rqarray) )
       {
       $value = shift(@rqarray);

       if ( $rqpairs{$key} ne "" )
          {
          $rqpairs{$key} .= "," . $value;
          }
       else
          {
          $rqpairs{$key} = $value;
          }
       }
}
sub url_decode {

#       Decode a URL encoded string or array of strings
#               + -> space
#               %xx -> character xx

    foreach (@_) {
        tr/+/ /;
        s/%(..)/pack("c",hex($1))/ge;
    }
    @_;
}
sub html_header {
    # Subroutine html_header sends to Standard Output the
    # necessary material to form an HHTML header for the
    #document to be returned, the single argument is the
    #TITLE field.
    local($title) = @_;
    #added the print http statement since the browser
    #keeps trying to download this script
    print "HTTP/1.0 200 OK\n";
    print "Content-Type: text/html\n\n";
    print "<html><head>\n";
    print "<title>$title</title>\n";
    print "</head>\n<body>\n";
}
sub html_trailer {
    # subroutine html_trailer sends the trailing material
    #to the HTML # on STDOUT.
    local($sec, $min, $hour, $mday, $mon, $year, $wday,
$yday, $isdst)
        = gmtime;
    local($mname) = ("Jan", "Feb", "Mar", "Apr", "May",
"Jun", "Jul",
                     "Aug", "Sep", "Oct", "Nov",
"Dec")[$mon];
    local($dname) = ("Sun", "Mon", "Tue", "Wed", "Thu",
"Fri",
                     "Sat")[$wday];

    print "<p>\nGenerated by: <var>$0</var><br>\n";
    print "Date: $hour:$min:$sec UT on $dname $mday
$mname $year.<p>\n";
    print "</body></html>\n";
}
#=====================================================================
#  ** USER MODIFICATIONS - BEGIN HERE **
#
#  Note: you should not need to modify anything above
# this line Your code should following this line.
#
#=====================================================================
sub error {
    # subroutine error sends an HTML error page to
STDOUT.
    local($msg) = @_;
    &html_header("SAS CGI Process Error");
    print "<H1>SAS CGI Process Error</H1>\n$msg\n";
    &html_trailer;
    exit 1;
}
# This is the directory SAS will write its temporary #
files to; must be writable by the web server's userid AND
# This is the directory that contains the .SAS files to
# run.
$PROGROOT="d:/website/htdocs/sugi22";
# This is the full path name of the SAS System.
$SAS611="d:/sas/sas.exe";

# This contains any special options you need to pass to
SAS.
# The -noxcmd option disallows X commands in the SAS
program.
# COMMENT OUT THIS OPTION SINCE IT's A UNIX command
#$C611="-noxcmd";

# Execute get_request subroutine

&get_request;

# Create unique name using process id and time
$pname="p$$".time;
# Output %LET statements to create SAS macro variables
# from the NAME=VALUE pairs.
open(OUTFI,">$PROGROOT/$pname.sas");
while ( ($name,$value) = each %rqpairs )
{
   # By using nrstr, special characters like semicolons
   # are allowed in the string. But we still have to
   # escape a few characters.
   $value =~ s/([%()'"])/%$1/g;
   print OUTFI "%let $name = %nrstr($value);\n";
}

print OUTFI "libname demo
'd:\\website\\htdocs\\sugi22';\n";
print OUTFI "%let outfl = %sysget(outfl);\n";
print OUTFI "options ls=80 nocenter nodate nonumber
mprint macrogen symbolgen;\n";
```

```
print OUTFI "title \"&username, Here are the Resources
You Requested!\";\n";
print OUTFI "footnote '(Data from Resource.sd2)';\n";
print OUTFI "%let where=;\n";
print OUTFI "%macro subsetit;\n";
print OUTFI "  %if \"&categcde\" ne \"ALL\" %then
%do;\n";
print OUTFI "     %let cat=\"&categcde\";\n";
print OUTFI "  %end;\n";
print OUTFI "  %else %let cat=;\n";
print OUTFI "  %if \"&type\" ne \"ALL\" %then %do;\n";
print OUTFI "     %let t=\"&type\";\n";
print OUTFI "  %end;\n";
print OUTFI "  %else %let t=;\n";
print OUTFI "  %if &cat ne and &t ne %then %let
where=where=categcde=&cat and type=&t,;\n";
print OUTFI "  %else %if &cat ne %then %let
where=where=categcde=&cat,;\n";
print OUTFI "  %else %if &t ne %then %let
where=where=type=&t,;\n";
print OUTFI "%mend;\n";
print OUTFI "%subsetit\n";
#Now let's decode the categcde field from devedu.html by
#using the category datafile to extract categdsc into
macro var catd
print OUTFI "data _null_;\n";
print OUTFI "  set demo.category;\n";
print OUTFI "  where \"&categcde\"=categcde;\n";
print OUTFI "  call symput(\"catd\",trim(categdsc));\n";
print OUTFI "run;\n";
#For the case when categcde="ALL"
print OUTFI "data _null_;\n";
print OUTFI "  if \"&categcde\"=\"ALL\" then call
symput(\"catd\",trim(\"ALL\"));\n";
print OUTFI "run;\n";

# USe encode=N in order to have hypertext links in my
tables
print OUTFI "%ds2htm(htmlfile=$PROGROOT/$pname.html,\n";
print OUTFI "          encode=N,\n";
print OUTFI "          bgtype=color,\n";
print OUTFI "          bg=white,\n";
print OUTFI "          brtitle=Results of &username
Query,\n";
print OUTFI "          ctext=red,\n";
print OUTFI "          tbbgcolr=yellow,\n";
print OUTFI "          obgcolr=white,\n";
print OUTFI "          clbgcolr=pink,\n";
print OUTFI "          csize=+2,\n";
print OUTFI "          openmode=replace,\n";
print OUTFI "          data=demo.resource,\n";
print OUTFI "          obsnum=y,\n";
print OUTFI "          &where\n";
print OUTFI "          var=name htmlfile synopsis,\n";
print OUTFI "          caption=Resources where Business
Category=&catd and Resource Type=&type);\n";
print OUTFI "endsas;";
close(OUTFI);

# Invoke SAS
#
system("$SAS611 $OPTIONS -sysin $PROGROOT/$pname.sas
                         -log $PROGROOT/$pname.log
                         -print $PROGROOT/$pname.lst
                         -sasuser $PROGROOT
                         -set outfl $PROGROOT/$pname.html
      ");

##############################################################
###############
# Here we print the resultant SAS output file.
# The output is now sent to the requesting
# web browser with HTML in it.
##############################################################
###############
print "HTTP/1.0 200 OK\n";
print "Content-Type: text/html\n\n";
open (FILE, "$PROGROOT/$pname.html") || die("Could not
find HTML file: $!\n");
while (<FILE>) {
        print;
}
close (FILE);

# Remove temporary files
# Comment these out when debugging.

unlink "$PROGROOT/$pname.sas";
unlink "$PROGROOT/$pname.html";
unlink "$PROGROOT/$pname.log";
exit;
```

The HTML Data Formatting Tool - D2HTM.SAS:

After downloading the HTML Data Formatting Tool from the SAS Institute's website, it was installed in a matter of minutes. As written on the website, "The HTML Data Set Formatter is an experimental tool provided by SAS Institute. The Data Set Formatter enables you to present any SAS data set as an HTML-formatted table. All you need is the Data Set Formatter macro and a Web browser capable of displaying tables."

Not only does the SUGI22.PL script write the SAS code; it also 'executes' pxxxx.sas and generates the pxxxx.log SAS log file. The D2HTM macro outputs HTML which is written to pxxxx.html as directed in the following statement:

```
print OUTFI
"%ds2htm(htmlfile=$PROGROOT/$pname.html,\n";
```

The syntax for the HTML Data Set Formatter is:

```
%DS2HTM(argument=value, argument=value,...);
```

There's a myriad of parameters that may be passed which enable you to control the presentation of your SAS data set to WWW clients. Here, I will only those parameters which were used in this application.

htmlfile=external-filename
specifies the name of the HTML file where the formatted output will be written. If the file you specify does not exist, it is created for you.

encode=Y | N
specifies whether the Data Set Formatter replaces angle brackets with the appropriate ASCII character representation so that the brackets display in the browser. To have the Data Set Formatter check for the characters "<" and ">" and encode them as ASCII characters, select Y. This will display the actual brackets in the browser. By default, the brackets will be encoded. To not encode them and have the Data Set Formatter pass the brackets to the browser (where the browser will attempt to act on them as an HTML-formatting instruction), select N.

Since the HTML links to the resources are stored in the Resource data table, I used encode=N. In this, way, these links will be displayed as 'clickable' hypertext links.

A future enhancement to the tool could be the addition of a parameter (or two) which would allow the developer to specify the table columns she would like to display as hypertext links. Therefore, rather than storing the HTMLFILE field in the Resource data table as <a href="../sugi22/book1.html">book1.html</a>, one would simply store ../sugi22/book1.html as the HTMLPATH and book1.html as HTMLFILE. The HTML Formatting Tool would provide the HTML tag appropriately.

At face value, one can argue that no savings are derived from this enhancement. But if you are dealing with gigabytes or even megabytes of data, you would want to minimize the storage of unnecessary characters in your database.

bgtype=NONE | COLOR | IMAGE
specifies the type of background for your Web page. Since I specified COLOR as the value, it was mandatory that I also use the BG argument. COLOR - causes the Data Set Formatter to use the background color specified in the BG argument.

bg=value
See bgtype above. In my call to ds2htm, I specified white as the background color.

brtitle=value
specifies the value that appears as the title in the browser window title bar. By default, no title is displayed. For this application, the browser title will be "Results of &username Query" where &username is resolved to be the username entered by the user.

ctext=value | DEFAULT
specifies whether you use the default global text color defined by the browser or by the color specified here. Here, all text will be red.

tbbgcolr=value | DEFAULT
specifies a background color for the entire table. The table background color is defined as yellow.

obgcolr=value | DEFAULT
specifies a background color for the column that contains observation numbers. This column will be white as indicated by the passed parameter.

clbgcolr=value | DEFAULT
specifies a background color for the column headers (column labels). The column headers cells will be pink.

csize=value | DEFAULT
specifies the size of the font used to display the caption text. Here a +2 was specified which is a 'relative' font size depending upon the browser's default font size.

openmode=APPEND | REPLACE
indicates whether the new HTML output overwrites the information currently in the specified file or if the new output is appended to the end of the existing file. For this application, REPLACE was selected.

data=SAS-data-set-name
specifies the SAS data set that you want to format using the Data Set Formatter. If you omit this argument, the Data Set Formatter will use the most recently created SAS data set. Demo.resource is the data set which the HTML formatter will process.

obsnum=Y | N
indicates whether the column containing observation numbers should be included in the table output. By default, the observation numbers are not included. In this application, the observation numbers will be displayed.

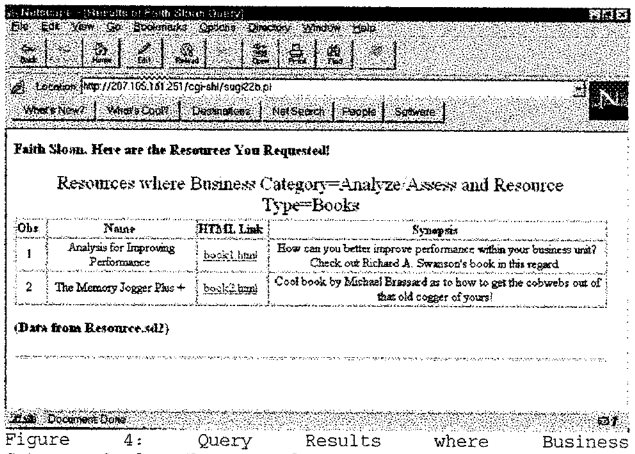
where=where-expression
specifies a valid WHERE clause that selects observations from the SAS data set. The &where macro variable will EITHER be where=categcde=&cat and type=&t, OR where=categcde=&cat OR where=type=&t, OR a null string. It was necessary to populate the &where macro variable with the complete where clause since the user may not want ANY subsetting at (in the case where categcde="ALL" and type="ALL"). If this is indeed the case, then &where resolves to a null string.

var=var1 var2 ...
specifies the variables that you want included in the HTML file and the order in which they should be included. To include all of the variables in the data set, do not specify the argument. Do not use a comma in the list of variable names.

caption=value
specifies the text that appears in the table caption. "Resources where Business Category=&catd and Resource Type=&type" is the caption where &catd and &type are resolved depending upon the values entered by the web user.

Query Results

If the educator selects 'Analyze/Assess' as the business category and 'Books' as the resource type and finally clicks the 'Submit this request' button, Figure 4 is displayed via the client's web browser:


File Edit View Go Bookmarks Options Directory Window Help
Location: http://207.105.161.251/cgi-shl/sugi22b.pl
What's New? What's Cool? Destinations Net Search People Software
Document Done

Faith Sloan. Here are the Resources You Requested!

Resources where Business Category=Analyze/Assess and Resource Type=Books

Obs	Name	HTML Link	Synopsis
1	Analysis for Improving Performance	book1.html	How can you better improve performance within your business unit? Check out Richard A. Swanson's book in this regard
2	The Memory Jogger Plus +	book2.html	Cool book by Michael Brassard as to how to get the cobwebs out of that old cogger of yours!

(Data from Resource.sd2)

Figure 4: Query Results where Business Category=Analyze/Access and Resource Type=Books

It provides the educator with a list of all available resources where business category="Analyze/Assess" and resource type="Books".

Conclusion

The development of the Developing Educators Resource World Wide Web Database System application met all of the primary system requirements:

1. **Use an existing personal computer**
2. **Use in-house software.**
3. **Use freely available tools and software to the extent possible.**
4. **Minimize the amount of programming required**
5. **Demonstrate the ability to obtain educators resources based on user-specified input.**
6. **Have fun!**

It took about 8 hours; it was inexpensive; it's practical; it works; and most importantly, it was fun!!

The SAS System in conjunction with a Web server, and the PERL programming language provided all the tools needed to develop this system.

A future enhancement of this application would be to allow the user to make 'multiple' business categories and resource types selections. Another enhancement would be to implement a boolean search mechanism which would allow the user to search for strings within the resource name and/or the resource synopsis.

Journey over to http://www.sas.com and check out their wonderful web tools!!

CONTACT:

FRS Associates
2750 Market Street, Suite 101
San Francisco, CA 94114-1987
http://www.frsa.com/
faith@frsa.com