Paper 154-25

Introduction to Dynamic Web Publishing

Faith R. Sloan, President/CEO
FRS Associates, LLC
San Francisco, CA 94114-1987

ABSTRACT

The World Wide Web is rapidly becoming the client of choice in enterprise systems. Web applications provide universal access, easy deployment, and low maintenance costs. This paper introduces you to the technologies and tools you can use to create interactive Web pages that access databases. It will begin with an overview of HTML, SAS HTML formatting tools, Perl CGI scripts and progresses through to a discussion of a full-fledge application.

Hypertext Markup Language (HTML)

This section briefly introduces you to the art of creating web pages the old-fashioned way -- by hand. There are many software "tools" that allow you create web pages without touching any HTML. You have to still correct the HTML editors' assumptions so you still have to become familiar with HTML. Also, if you are serious about doing more than a simple page or two, and want to further your knowledge as well as your market value, a solid footing in the basics will greatly accelerate what you can do and command.

Simply stated, HTML, or HyperText Markup Language, is a page layout language that tells a web browser how to display a web page. It doesn't even come close to being a programming language. It's rather simple for the most part. The documents themselves are plain text files with special "tags" or codes that a web browser knows how to interpret and display on your screen.

Now that you know what HTML is, let's start using it.

An HTML document contains two distinct parts: the head and the body. The head contains information about the document
that is not displayed on the screen. The body then contains everything else that is displayed as part of the web page.

The basic structure then of any HTML page is:

```
<HTML>
  <HEAD>
  <!-- header info used to contain extra information
about
       this document, not displayed on the page -->
  </HEAD>
  <BODY>
   <!-- all the HTML for display -->
          :      :
          :      :
          :      :
  </BODY>
  </HTML>
```

With this in mind, let's create our first web page using HTML.

```
<HTML>
  <HEAD>
    <TITLE>SAS TIPS and Techniques</TITLE>
  </HEAD>
  <!-- Faith R. Sloan 15February1998 SAS TIPS and
Techniques-->
  <!—Copyright by FRS Associates, LLC 1998 All rights
Reserved-->
  <BODY>
        "SAS Tips are in abundance on the web. You just
        have to find them. Check out <a href=" http:
        //www.frsa.com/sasmass</a> for weekly tips,
        techniques, and more.
  </BODY>
  </HTML>
```

Notice where the <title>...</title> tags are located. They are in the <head>...</head> section and thus will not be visible in the browser. The <title> tag is used to uniquely identify each document and is also displayed in the title bar of the browser window. This is what the web-based search engines will usually pick up to describe the document, so make it self-explanatory.

Also note that there is a comment tag <!-- ...--> that lists the name of the author and the date the document was created. You could write anything in between the comment tags but it is only visible when you view the actual source of the web page.

Now that you've created your first web page, how can you view it?

1. Save the file as index.html
2. Open your web browser
3. Select Open File... from the File menu.
4. Use the dialog box to find and open the file you created, "index.html"
5. You should now see in the title bar of the workspace window the text "SAS Tips and Techniques" and in the web page below, the one sentence of <body> text you wrote, 'SAS Tips are in abundance on the web. You just have to find them. Check out SAS.Masses for weekly tips, techniques, and more.'

You have created your first web page! The use of an HTML anchor <a href= ...> was used in order to create the clickable link attached to SAS.Masses.

There are many web-based tutorials and primers online. One that I highly recommend is The NCSA's Beginner's Guide to HTML at http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html.

HTML Forms and Common Gateway Interface (CGI)

An HTML fill-in form is implemented by using the HTML FORM element (or tag). Here I will demonstrate a simple example so that you are able to follow the example in the 'Putting It All Together' section.

The FORM tag is used to create fill-in forms with input and select elements. These may include checkboxes, radio buttons, text boxes, and submit buttons. The form simply collects the data but does not process it. The data from the form is then sent to a server-side gateway program for processing.

The following example uses just two of the available FORM elements -- namely, an INPUT element with an attribute of the text TYPE; a SELECT element which allows for selecting from a pull-down list; and an INPUT element with an attribute of the submit TYPE.

The most important attribute to the FORM elements is NAME, which assigns a 'variable name' to the 'value' entered into the element. When the user presses the "Submit this request" SUBMIT button, the values of the FORM fields are encoded and sent to the server along with other 'environment' variables and 'headers' that I won't discuss here. Since we specified "METHOD=POST", the data are sent to the gateway program, http://207.105.181.251/cgi-shl/getResource.pl as an input stream which the program reads from standard input. The getResource.pl perl CGI program receives then parses the FORM data by using the variable names to interpret the contents of the corresponding values entered by the user in order to separate the fields. Don't get nervous about writing code to perform this parsing. There are free perl libraries downloadable from the web that does it for you. All you need to do is to reference the libraries from within your perl program. Finally, getResource.pl is executed on the server!

```
<FORM ACTION="http://207.105.181.251/cgi-
shl/getResource.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
</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>
<INPUT TYPE=submit VALUE="Submit this request">
</FORM>
```

The Common Gateway Interface (CGI) is simply the specified standard for communications between HTTP (web) servers and server-side gateway programs. When you access the gateway program, the server activates the program and passes it any FORM or other data that was sent from the client (browser). When the gateway program has finished processing the data, it sends the result back to the server, where the server sends it back to the client. The results could be something as simple as "Faith, here are the results of your search!" along with a listing of search results displayed in my client browser. In the previous example the gateway perl program that I wrote is getResource.pl stored on the web server in the cgi-shl directory.

Gateway programs can be compiled programs written in such languages as C, C++, Pascal, or they can be executable scripts written in languages such as perl, tcl, and various shell programs. Most CGI programs are scripts, since these are easy to write and modify and are more portable than compiled programs which are platform specific.

Now, let's talk about what SAS Institute Inc brings to the table before we put it all together.

SAS Web Publishing Tools

The SAS Institute offers free, downloadable Web Publishing Tools from their web site at http://www.sas.com/web. Three of these tools fall under the realm of *HTML Formatting Tools*. They are a collection of SAS macros that enable you to display your SAS data sets and procedure output via the web without you having to learn HTML. But of course, I recommend that you learn HTML anyway.

There are three HTML Formatting Tools or SAS macros:

- **Data Set Formatter - %ds2htm**
- **Output Formatter - %out2htm**
- **Tabulate Formatter - %tab2htm**

The next three sections will go into more detail as to how these three HTML formatting tools are utilized.

Dataset Formatter - %ds2htm

The dataset formatter macro generates an HTML-formatted document from any SAS dataset.

This macro is powerful in that it supports WHERE clauses, BY-group processing, and other data presentation capabilities.

The following example demonstrates the simplicity in which this macro works:

```
LIBNAME MYLIB "D:\WEBSITE\HTDOCS\SUGI25";
FILENAME OUTIT "D:\WEBSITE\HTDOCS\SUGI25\dataset.html";
OPTIONS LS=80;
%DS2HTM(htmlfref=outit,
        openmode=replace,
        data=MYLIB.resource,
        bgtype=color,
        bg=white,
        brtitle=Subsetted Resource Dataset,
        ctext=red,
        tbbgcolr=yellow,
        clbgcolr=pink,
        csize=+2,
        obsnum=n,
        where=categcde eq ”AA”,
        var=name htmlfile synopsis,
        caption=Resources where Business
Category=”Analyze and Assess);
```

The above code calls the DS2HTM macro and passes a cornucopia of parameters.

The HTML output generated from the macro is written to the filename referenced by OUTIT, D:\WEBSITE\HTDOCS\SUGI25\dataset.html.

The OPENMODE parameter is set to 'replace' which ensures that a new file is created anytime the macro is executed. Another option is to use 'append' which comes in handy if:

1. You have multiple HTML formatting macros at various stages within your SAS source code and you want to append to the html output file.
2. You want to append to an existing html output file

The SAS dataset that will be formatted is MYLIB.resource.

BGTYPE specifies the type of background for Web page. Since I used COLOR as the value, I must also use the BG argument that I set to white. The text color is specified by CTEXT to be red; the table background color is defined by TBBGCOLR which is yellow; the column header background color is pink; the size of the text is increased relatively by a factor of 2 as specified by CSIZE; and OBSNUM=n ensures that the observation numbers are not displayed. The WHERE parameter allows for subsetting; the VAR attribute specifies the variables values and headers which will be outputted; and finally the CAPTION will be placed at the top of the table.

The resulting output is displayed in the browser as follow:

The SAS System

Resources where Business Category=Analyze and Assess

Name	HTML Link	Synopsis
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.
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!
Need a Laugh??	Laugh.html	Need to relieve a bit of stresS? This resource will truly have you in stitches!! Try it out!
Resource Name c da da da	Develop2.pdf	yet another model

The HTML generated and written to dataset.html is:

```
<HTML>
<HEAD>
  <META    NAME="GENERATOR"
        CONTENT="SAS Institute Inc. HTML Formatting
Tools, http://www.sas.com/">
  <TITLE>Subsetted Resource Dataset</TITLE>
</HEAD>
<BODY BGCOLOR=white TEXT=red>
<PRE><H3>The SAS System</H3></PRE>
<P>
<TABLE BORDER=1 WIDTH=100% ALIGN=CENTER CELLPADDING=1
CELLSPACING=1 BGCOLOR=yellow>
  <CAPTION ALIGN=CENTER VALIGN=TOP><FONT
SIZE=+2>Resources where Business Category=Analyze and
Assess</FONT></CAPTION>
  <TR>
    <TH BGCOLOR=pink ALIGN=CENTER VALIGN=MIDDLE>Name
</TH>
    <TH BGCOLOR=pink ALIGN=CENTER VALIGN=MIDDLE>HTML
Link    </TH>
    <TH BGCOLOR=pink ALIGN=CENTER VALIGN=MIDDLE>Synopsis
   </TH>
  </TR>
  <TR>
    <TD ALIGN=CENTER VALIGN=MIDDLE>Analysis for
Improving Performance    </TD>
    <TD ALIGN=CENTER VALIGN=MIDDLE><a
href="../SUGI25/book1.html">book1.html</a>    </TD>
    <TD ALIGN=CENTER VALIGN=MIDDLE>How can you better
improve performance within your business unit? Check out
Richard A. Swanson's book in this regard.    </TD>
  </TR>
  <TR>
    <TD ALIGN=CENTER VALIGN=MIDDLE>The Memory Jogger
Plus +    </TD>
    <TD ALIGN=CENTER VALIGN=MIDDLE><a
href="../SUGI25/book2.html">book2.html</a>    </TD>
    <TD ALIGN=CENTER VALIGN=MIDDLE>Cool book by Michael
Brassard as to how to get the cobwebs out of that old
cogger of yours!    </TD>
  </TR>
  <TR>
    <TD ALIGN=CENTER VALIGN=MIDDLE>Need a Laugh??
</TD>
    <TD ALIGN=CENTER VALIGN=MIDDLE><a
href="../SUGI25/laugh.html">laugh.html</a>    </TD>
    <TD ALIGN=CENTER VALIGN=MIDDLE>Need to relieve a bit
of stresS? This resource will truly have you in
stitches!! Try it out!    </TD>
  </TR>
  <TR>
    <TD ALIGN=CENTER VALIGN=MIDDLE>Resource Name c da da
da    </TD>
    <TD ALIGN=CENTER VALIGN=MIDDLE><a
href="../SUGI25/develop2.pdf">develop2.pdf</a>    </TD>
    <TD ALIGN=CENTER VALIGN=MIDDLE>yet another stinkin'
model    </TD>
  </TR>
</TABLE>
<P>
<HR>
</BODY>
</HTML>
```

Output Formatter - %out2htm

The output formatter macro saves output from any SAS procedure to an HTML file. It converts SAS procedure output to HTML tables.

The following example demonstrates how to capture the output from the CONTENTS procedure:

```
LIBNAME MYLIB "D:\WEBSITE\HTDOCS\SUGI25";
FILENAME OUTIT "D:\WEBSITE\HTDOCS\SUGI25\contents.html";
OPTIONS LS=80;
%OUT2HTM(capture=on,
         window=output,
         runmode=b);
  PROC CONTENTS DATA=MYLIB.RESOURCE;
  RUN;
%OUT2HTM(htmlfref=outit,
         capture=off,
         window=output,
         openmode=replace,
         runmode=b,
         tcolor=red,
         tface="Arial,Helvetica",
         hcolor=green);
RUN;
```

The above code calls the OUT2HTM macro and turns capture mode to 'on' for the output window. What this means is capture the output from the output window of all procedures until we encounter another call to the OUT2HTM macro which has capture mode=off.

The output from the CONTENTS procedure is captured and is output to the filename referenced by OUTIT, D:\WEBSITE\HTDOCS\SUGI25\contents.html.

The OPENMODE parameter is set to 'replace' which operates just as it does with the DS2HTM macro.

The RUNMODE parameter is set to 'b' which means that this macro is being executed in batch mode.

TCOLOR here will render all title lines using the color red while HCOLOR will render all header lines using the color green. TFACE allows us to choose a font face for the title lines. Our first choice is "Arial" and if the user doesn't have that particular font face on their computer, then "Helvetica" is used. If neither of these font faces exists on the user's computer, then the default specified within their browser is used instead.

The resulting output is displayed in the browser as follow:

```
The SAS System                                1
                                                                    23:47
Thursday, February 10, 2000

                              CONTENTS PROCEDURE

Data Set Name: MYLIB.RESOURCE
Observations:          49
Member Type:   DATA
Variables:             5
Engine:        V612
Indexes:               0
Created:       2:04 Monday, January 6, 1997
Observation Length:   380
Last Modified: 0:29 Tuesday, January 7, 1997
Deleted Observations: 0
Protection:
Compressed:            NO
Data Set Type:
Sorted:                NO
Label:

                  -----Engine/Host Dependent
Information-----

                       Data Set Page Size:       11776
                       Number of Data Set Pages: 2
                       File Format:              607
                       First Data Page:          1
                       Max Obs per Page:         30
                       Obs in First Data Page:   28

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

               #    Variable    Type    Len     Pos
Label

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

               3    CATEGCDE    Char       5      75
Category Code
               5    HTMLFILE    Char     100     280
HTML Link
               1    NAME        Char      60       0
Name
               4    SYNOPSIS    Char     200      80
Synopsis
               2    TYPE        Char      15      60
Type
```

where
The SAS System 1 23:47
Thursday, February 10, 2000
is red and the headers such as CONTENTS PROCEDURE, -----Engine/Host Dependent Information-----, etc are all in green.

The resulting HTML is as follow:

```
<HTML>
<HEAD>
  <META    NAME="GENERATOR"
       CONTENT="SAS Institute Inc. HTML Formatting
Tools, http://www.sas.com/">
</HEAD>
<BODY>
<PRE><H3><FONT FACE="Arial,Helvetica" COLOR=red>
                         The SAS System
               1
                                                        23:47
Thursday, February 10, 2000
</FONT></H3></PRE>
<PRE><STRONG><FONT COLOR=green>
      CONTENTS PROCEDURE
</FONT></STRONG></PRE>
<PRE>Data Set Name: MYLIB.RESOURCE
   Observations:          49
Member Type:   DATA
Variables:             5
Engine:        V612
Indexes:               0
Created:       2:04 Monday, January 6, 1997
Observation Length:   380
Last Modified: 0:29 Tuesday, January 7, 1997
Deleted Observations: 0
Protection:
Compressed:            NO
Data Set Type:
Sorted:                NO
Label:
</PRE>
<PRE><STRONG><FONT COLOR=green>                    -----
Engine/Host Dependent Information-----
</FONT></STRONG></PRE>
<PRE>
                       Data Set Page Size:       11776
                       Number of Data Set Pages: 2
                       File Format:              607
                       First Data Page:          1
                       Max Obs per Page:         30
                       Obs in First Data Page:   28
</PRE>
<PRE><STRONG><FONT COLOR=green>               -----
Alphabetic List of Variables and Attributes-----

               #    Variable    Type    Len     Pos
Label

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ</FO
NT></STRONG></PRE>
<PRE>               3    CATEGCDE    Char       5      75
  Category Code
               5    HTMLFILE    Char     100     280
HTML Link
               1    NAME        Char      60       0
Name
               4    SYNOPSIS    Char     200      80
Synopsis
               2    TYPE        Char      15      60
Type</PRE>
<HR>
</BODY>
</HTML>
```

Notice the font color and font face attributes within the HTML document above.

Tabulate Formatter - %tab2htm

The tabulate formatter macro saves output from the SAS TABULATE procedure to an HTML file.

The following example demonstrates how to capture the output from the TABULATE procedure:

```
LIBNAME MYLIB "D:\WEBSITE\HTDOCS\SUGI25";
FILENAME OUTIT "D:\WEBSITE\HTDOCS\SUGI25\tabulate.html";
OPTIONS LS=80 NODATE NONUMBER;
```

```
PROC SORT DATA=MYLIB.RESOURCE OUT=RES;
  BY CATEGCDE;
RUN;
PROC SORT DATA=MYLIB.CATEGORY OUT=CAT;
  BY CATEGCDE;
RUN;

DATA NEWRES;
  MERGE RES(in=r) CAT; /* Decodes for categories */
  BY CATEGCDE;
  IF R;
RUN;
%TAB2HTM(capture=on, runmode=b);
  PROC TABULATE data=NEWRES format=comma10.
formchar='82838485868788898a8b8c'x;
   CLASS categdsc type;
   TABLE type*categdsc;
   LABEL categdsc='Category';
   TITLE "TAB2HTM Example - WUSS 98";
  RUN;
%TAB2HTM(htmlfref=outit,
         capture=off,
         openmode=replace,
         runmode=b,
         clcolor=red,
         twidth=80);
RUN;
```

The above code calls the TAB2HTM macro and turns capture mode to 'on' in the same manner as the OUT2HTM macro works with the exception of the WINDOW parameter.

The output from the TABULATE procedure is captured and is output to the filename, D:\WEBSITE\HTDOCS\SUGI25\tabulate.html referenced by OUTIT.

The OPENMODE and RUNMODE parameters operate in the same manner here as in the OUT2HTM macro as well. The CLCOLOR parameter defines the column color to be red. Finally, TWIDTH defines the width of the HTML table as 80.

The resulting output from the TABULATE procedure is displayed in the browser and the actual generated HTML document, tabulate.html, contains the following:

```
<HTML>
<HEAD>
  <META NAME="GENERATOR"
        CONTENT="SAS Institute Inc. HTML Formatting
Tools, http://www.sas.com/">
</HEAD>
<BODY>
<PRE><H3>TAB2HTM Example - WUSS 98</H3></PRE>
<PRE><H3></H3></PRE>
<P>
<TABLE BORDER=1 WIDTH=80% ALIGN=CENTER CELLPADDING=1
CELLSPACING=1>
  <TR>
    <TH ALIGN=CENTER VALIGN=BOTTOM COLSPAN=7
NOWRAP><FONT COLOR=red>Type</FONT></TH>
  </TR>
  <TR>
    <TH ALIGN=CENTER VALIGN=BOTTOM COLSPAN=7
NOWRAP><FONT COLOR=red>Books</FONT></TH>
  </TR>
  <TR>
    <TH ALIGN=CENTER VALIGN=BOTTOM COLSPAN=7
NOWRAP><FONT COLOR=red>Category</FONT></TH>
  </TR>
  <TR>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>Accelerat-<BR>ing<BR>Learning,<BR>Developme-
<BR>nt, &<BR>Performan-<BR>ce</FONT></TH>

    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>Achieving<BR>Business<BR>Results</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>Alternati-
<BR>ve<BR>Learning<BR>Solutions</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>Analyze/A-<BR>ssess</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>Change<BR>Management</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>Consult</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>Consulting</FONT></TH>
  </TR>
  <TR>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>N</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>N</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>N</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>N</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>N</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>N</FONT></TH>
    <TH ALIGN=CENTER VALIGN=BOTTOM NOWRAP><FONT
COLOR=red>N</FONT></TH>
  </TR>
  <TR>
    <TD ALIGN=RIGHT VALIGN=BOTTOM NOWRAP>1</TD>
    <TD ALIGN=RIGHT VALIGN=BOTTOM NOWRAP>1</TD>
    <TD ALIGN=RIGHT VALIGN=BOTTOM NOWRAP>5</TD>
    <TD ALIGN=RIGHT VALIGN=BOTTOM NOWRAP>2</TD>
    <TD ALIGN=RIGHT VALIGN=BOTTOM NOWRAP>1</TD>
    <TD ALIGN=RIGHT VALIGN=BOTTOM NOWRAP>4</TD>
    <TD ALIGN=RIGHT VALIGN=BOTTOM NOWRAP>4</TD>
  </TR>
</TABLE>
<P>
<B><I> (CONTINUED)</I></B><P>
<HR><P>
<PRE><H3>TAB2HTM Example - WUSS 98</H3></PRE>
<PRE><H3></H3></PRE>
<P>
<TABLE BORDER=1 WIDTH=80% ALIGN=CENTER CELLPADDING=1
CELLSPACING=1>
.
.
.
</TABLE>
<P>
<HR>
<P>
</BODY>
</HTML>
```

Putting It All Together

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='../SUGI25/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/getResource.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/SUGI25/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
```

Figure 2: Category Table Structure

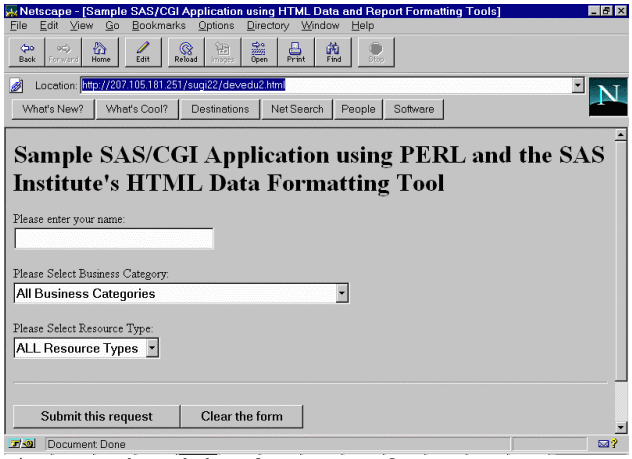

Figure 3. The Web-based user Interface

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 getResource.pl PERL script. This is referenced in the HTML FORM as <FORM ACTION="http://207.105.181.251/cgi-shl/getResource.pl" METHOD=POST>. The three parameters which will be passed to getResource.pl to operate upon are name, categcde, and type.

Now we are ready to discuss the intricacies of the getResource.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 - getResource.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 getResource.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/SUGI25";
# 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\\SUGI25';\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 getResource.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. Consult the SAS documentation which is available when you download the Web Publishing tools from http://www.sas.com/web.

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:

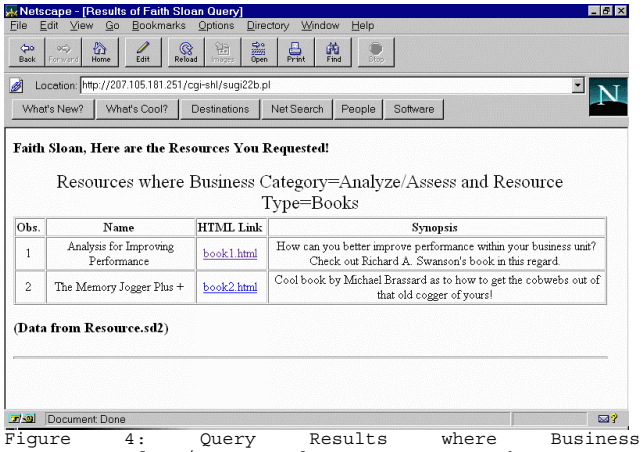

Figure 4: Query Results where Business Category=Analyze/Access and Resource Type=Books

It provides the educator with a list of all available resources where business

category="Analyze/Assess" and resource type="Books".

Conclusion

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

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.

I would highly recommend that you venture on over to the SAS Institute's web site at http://www.sas.com/web to see what else they have to offer in terms of web enablement using the SAS Institute's SAS/IntrNet Application Dispatcher component and more.

There are many also other players in the web scripting tools and web application server market such as Allaire's Cold Fusion product at http;//www.allaire.com, BEA's Web Logic, at http://www.beasys.com, and of course there's the Microsoft scripting tool, Active Server Pages (ASP) which is packaged with the Microsoft Internet Information Server (IIS - Web Server).

Happy computing!

CONTACT:

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