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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 NAME</td>
<td>Char</td>
<td>60</td>
<td>0</td>
<td>Name</td>
</tr>
<tr>
<td>2 TYPE</td>
<td>Char</td>
<td>15</td>
<td>60</td>
<td>Type</td>
</tr>
<tr>
<td>3 CATEGCD</td>
<td>Char</td>
<td>5</td>
<td>75</td>
<td>Category Code</td>
</tr>
<tr>
<td>4 SYNOPSIS</td>
<td>Char</td>
<td>200</td>
<td>80</td>
<td>Synopsis</td>
</tr>
<tr>
<td>5 HTMLFILE</td>
<td>Char</td>
<td>100</td>
<td>280</td>
<td>HTML Link</td>
</tr>
</tbody>
</table>

Sample Record:

Interactive by Design Books How to Design whatever link: book5.html

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.

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

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Type</th>
<th>Len</th>
<th>Pos</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CATEGORY</td>
<td>Char</td>
<td>50</td>
<td>Description</td>
</tr>
<tr>
<td>2</td>
<td>CATEGCOE</td>
<td>Char</td>
<td>5</td>
<td>Code</td>
</tr>
</tbody>
</table>

Sample Record:
Design/Create DC {PRIVATE

Figure 2: Category Table Structure

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://faith.hypno.net/sugi22/devedu.html and is illustrated in Figure 3.

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 faith.hypno.net to execute the SUGI22.PL PERL script (or program). This is referenced in the HTML FORM as <FORM ACTION='http://faith.hypno.net/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 one time, this ensures that a unique file will be
The d2htm macro is a SAS Institute supplied 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.

```perl
sub get_request {
    # Subroutine get_request reads the POST or GET form
    # request from SUBMIT into the variable $request, and
    # then splits it into its
    # name-value pairs in the associative array $reqpairs.
    # The number of bytes is given in the environment
    # variable CONTENT_LENGTH which is automatically set by the
    # request generator.
    if ($ENV{'REQUEST_METHOD'}) eq "POST" {
        read($ENV{'REQUEST_METHOD'}, $ENV{'CONTENT_LENGTH'});
    } elsif ($ENV{'REQUEST_METHOD'}) eq "GET" {
        $request = $ENV{'QUERY_STRING'};
    }

    $reqpairs = ();
    $reqarray = url_decode(splith%/\$1/, $request);
    while ( ($key = shift($reqarray)) ) {
        $value = shift($reqarray);
        if ( $reqpairs($key) ne "")
            $reqpairs($key) .= "," . $value;
        else
            $reqpairs($key) = $value;
    }
}

sub url_decode {
    # Decode a URL encoded string or array of strings
    + => space
    %- => character xx
    foreach (@_) {
        $tx=./;
        s/%(.)/pack("c",hex($1))/ge;
        @_;}
}

sub html_header {
    # Subroutine html_header sends to Standard Output the
    # necessary material to form an HTML header for the
    # document to be returned, the single argument is the
    # TITLE field.
    local($title) = 0;
}
```

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.

```perl
local($smame) = "p121852472586.sas,
local($iyear) = "p121852472586.log,
local($istart) = "p121852472586.html where $pname = p121852472586.

sub get_request {
    # Subroutine get_request reads the POST or GET form
    # request from SUBMIT into the variable $request, and
    # then splits it into its
    # name-value pairs in the associative array $reqpairs.
    # The number of bytes is given in the environment
    # variable CONTENT_LENGTH which is automatically set by the
    # request generator.
    if ($ENV{'REQUEST_METHOD'}) eq "POST" {
        read($ENV{'REQUEST_METHOD'}, $ENV{'CONTENT_LENGTH'});
    } elsif ($ENV{'REQUEST_METHOD'}) eq "GET" {
        $request = $ENV{'QUERY_STRING'};
    }

    $reqpairs = ();
    $reqarray = url_decode(splith%/\$1/, $request);
    while ( ($key = shift($reqarray)) ) {
        $value = shift($reqarray);
        if ( $reqpairs($key) ne "")
            $reqpairs($key) .= "," . $value;
        else
            $reqpairs($key) = $value;
    }
}
```

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.

```perl
local($smame) = "p121852472586.sas,
local($iyear) = "p121852472586.log,
local($istart) = "p121852472586.html where $pname = p121852472586.

```

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.
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:

```sas
print OUTFI "d:\website\htdocs\suqi22.html";
```

The syntax for the HTML Data Set Formatter is:

```sas
%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 use those parameters which were used in this application.

- `http://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-formating instruction), select N.

### The HTML Data Formatting Tool - D2HTM.SAS

Before running the SAS code, you may want to direct the output to a SAS log file. It is a good idea to also comment out the macro used to generate the HTML file and the macro used to create the example file. Also, the browser used to display the file will attempt to act on the `<` and `>` characters, which may cause problems if not encoded properly. If you choose not to encode these characters, you can still use the tool, but the output will not be displayed correctly in the browser.

```sas
/*web browser with HTML in it.*/
%INCLUDE "SUGI22.SAS";
%include "$PROGR\Root\$pname.html";
print "Content-Type: text/html";
open (FILE, "$PROGR\Root\$pname.html") | die ("Could not find HTML file: $!\n")
while (@FILE) {
  print;
}
close (FILE);
```

### Remove temporary files

- `@FILE` syntax is used to remove temporary files.

- `@FILE` syntax is also used to comment these out when debugging.

```sas
unlink "$PROGR\Root\$pname.sas";
unlink "$PROGR\Root\$pname.html";
unlink "$PROGR\Root\$iname.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:

```sas
print OUTFI "d:\website\htdocs\suqi22.html";
```

The syntax for the HTML Data Set Formatter is:

```sas
%DS2HTM(argument=value, argument=value, ...);
```
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/bookl.html">bookl.html</a>, one would simply store ../sugi22/bookl.html as the HTMLPATH and bookl.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.

color=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.

olbgcolr=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:
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 educator 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.frs.com, faith@frs.com