Mass Producing Web Pages with SAS® Software

Keith J. Brown

Keith J. Brown is an applications analyst in the Program Assessment and Public Service division of the University of North Carolina General Administration. He has been using SAS® software since 1981 as a student, instructor, consultant, and analyst, and is currently president of the Research Triangle SAS Users Group. Keith's major areas of interest include SAS/GRAPH® software, efficient programming techniques, and using SAS software to build Web pages. He has a baccalaureate degree from the University of Alabama and a master's degree from the University of North Carolina at Chapel Hill, both in political science, and is active in a local community theater group.

Abstract

With the wide variety of products available today, producing a Web page literally has become child's play. Converting a report of several hundred pages into an equivalent number of Web pages, however, is still a herculean task. By using the SAS® System, this process can be automated, as well as the job of building an index Web page to guide the user to the appropriate report. In addition, the same program can still produce hardcopy output when needed. The methods shown here are most appropriate for relatively static data, where instant access to the most recent figures is not required.

Contents

- Introduction
- Setting the Parameters
- Building the Counters
- Building the Macros
- The Index Page Macro
- The PROC TABULATE Macro
- The Hardcopy Macro
- The Plain HTML Macro
- The Fancy HTML Macro
- The Big Loop Macro
- Conclusion
- References

Introduction

The invention of the printing press (and its relatives the typewriter and the computer printer) have made possible the greatest information distribution system the world has ever known -- bureaucracy. Every day, millions of pages of informative reports are carefully printed, collated, stapled, mailed, and then filed away where no one will ever see them again.

With the advent of the World Wide Web, this information can now be disseminated almost instantaneously to anyone with access to the Internet or an organization's intranet. Most importantly, people who need information can obtain it directly from those who have it.

One method that SAS software users have of making this information available is through SAS/IntrNet™ software, which allows a Web surfer with a "thin client" (a PC with just a Web browser) to use your Web server to submit programs that will run against your SAS databases and return HTML-formatted output to the browser's screen. This product allows the browser to access the most current information available.

While constantly changing data might require SAS/IntrNet software, many of us have data that is much more stable; we are more accustomed to producing reports at regular intervals from a number of different subgroups of our data. Instead of generating reports on the fly from warehoused data, we can produce large numbers of static Web pages once, and provide index pages as roadmaps to allow our users to view the specific report that they need. Mass producing this type of Web page is the focus of this presentation.

To illustrate this, imagine that you are a programmer at the fictitious University of Chatham. You've just handed your manager the output from a rather elaborate PROC TABULATE displaying the retention, graduation, and persistence rates of students over the past ten semesters. Your next assignment is to place that output on the World Wide Web, which you do with little difficulty. Having downloaded the HTML formatting tools from SAS Institute's Web site and installed them on your system, you upgrade your TABULATE output by using the %TAB2HTM macro.

Your boss is quite pleased with the result, as are the chancellors, deans, and department chairs of various institutions in the University of Chatham system. In fact, they are so pleased that each one wants a separate Web page for his or her department, division, or campus. You are now faced with the task of building 231 Web pages and building an index page so that anyone can easily find a given report.

Just to make things more interesting, the machine on which you will be running the SAS System is on the other side of a firewall from your organization's Web site. Of course, you have be prepared to rebuild the pages when you next receive new data, and you need to be able to generate printed reports to be collated, stapled, mailed, and filed, so that bureaucracy as we know it will prosper.

To accomplish these tasks, you will need to write several macros that can be invoked for different types of output (printed and Web) at different levels of analysis (campus, division, and department). Anything in your program that should change when it is re-run--dates, types of output, or subsetting variables--should be specified as a macro variable.

Setting the Parameters

Our sample program takes in three pieces of information from the SAS System option, SYSPARM--the current year, number of semesters to be tracked, and the type of output desired. The accompanying DATA step creates four macro variables, some of them derived from the SYSPARM values, others created as constants. These macro variables control the environment for a given execution of the program. When this program needs to be re-run, either because we have new data or we wish to change the type of output we generate, the only line of code that should need modification is the OPTIONS SYSPARM line.

The SYSPARM option creates an automatic macro variable, which can be referenced as &SYSPARM. Its contents can also be accessed through the SYSPARM() function, as shown in the code below. While we could also create the macro variables &YR1, &SPAN, and &TYPE by using %LET statements, using SYSPARM allows us to create additional macro variables by using a variety of familiar functions and statements. In addition, under many operating systems, you can specify a SYSPARM option at the time that you invoke SAS software for a particular execution.

```
************************************************************;
* Section 1. Setting Parameters. Input values: Current year, *;
*            no. of semesters to be tracked, and runtype     *;
*            (WEB1, WEB2, PRT1).                             *;
************************************************************;
options sysparm="1998,10,WEB2"    nodate nonumber
        ovp nobyline ls=165 ps=55 symbolgen nofmterr nosource2;

libname in  "/uchatham/planning";
libname library "/uchatham/fmts";
```

```
data _null_;
length blanks $ 100;
yr1=input(substr(sysparm(),1,4),4.);   * Current year of report *;
spn=input(substr(sysparm(),6,2),2.);   * Number of semesters being tracked *;
type=upcase(substr(sysparm(),9));      * Type of output desired *;
blanks=repeat(' ',99);                 * Left justify footnotes *;
call symput("yr1",compress(yr1));
call symput("span",compress(spn));
call symput("type",compress(type));
call symput("blanks",blanks);
run;
```

Building the Counters

The process of producing hundreds or even thousands of Web pages with appropriate index pages will make use of several features of the SAS System, some old (such as macros and PROC PRINTTO) and some new (such as the FTP file access method and HTML formatting tools).

While a BY statement is often sufficient to generate printed reports for different subgroups, more elaborate measures are required to generate multiple Web pages from a single iteration of a program. Output destined for each different Web page requires some sort of identifier. One of the first steps in adapting existing report-generating programs to produce Web pages is to build these identifier variables for each desired report.

In addition to generating the reports, we also need to generate an index Web page that will allow a Web surfer to select a given report with ease. Because our index Web page will contain links to reports down to the departmental level, the data set that generates that page needs to contain one observation for each department, along with information about the division and institution for that department.

We use the RETAIN statement, FIRST.variable, and LAST.variable logic to produce three data sets that contain the data needed to write the reports, build the index Web page, and run the macro that will write the reports out into the proper pages.

Because macros require index variables with equal intervals, we need to assign each institution, division, and department a sequential number (INSTNO, SCHLNO, and DEPTNO). For the index data set, we write one observation per department; for the macro counter data set, we need one observation per division.

```
*************************************************************;
* Section 2. Building the index and counter data sets.      *;
*************************************************************;

proc sort data=one;
by instid schlname deptname;
run;

data one
     index(keep=instid instno schlno  schlname deptno deptname)
     counter(keep=instno schlno deptno)
;
*****************************************************************;
* WORK.ONE contains the data to be displayed in PROC TABULATE.  *;
* WORK.INDEX has one observation per department and is used to  *;
*            build the links from the index Web page to the     *;
*            individual Web pages for each campus, division, and*;
*            department.                                        *;
* WORK.COUNTER has one observation per division and is used to  *;
*            control the number of times the %LOOP1 macro       *;
*            executes.                                          *;
*****************************************************************;
;
```

```
set one;
by instid schlname deptname;
retain instno 0 schlno 0  deptno 0;

   * Increment campus counter, set division and dept counters to 0 *;
   * for each new instutition.                                     *;
if first.instid then do; instno=instno+1; schlno=0; deptno=0; end;

   * Increment division counter, set dept counter to 0 for each    *;
   * college or school.                                           *;
if first.schlname then do; schlno=schlno+1; deptno=0; end;

   * Increment department counter for each new dept.              *;
if first.deptname then deptno=deptno+1;

output one;
if last.deptname then output index;
if last.schlname then output counter;
run;
```

Building the Macros

With these data sets created, the remainder of the program is a set of macros that can produce regular hardcopy output (%PRT1), very basic Web pages (%WEB1), or use HTML formatting tools for more elaborate Web output (%WEB2). All three of these macros invoke a single macro (%LISTING) that produces the PROC TABULATE output.

While the sample program here produces only a single report, there is no such limit imposed by necessity. Instead of a single PROC TABULATE in the macro %LISTING, you might have several procedures or customized reports, so long as their output is controlled by the same macro variable references.

If one of the Web options is selected, the program automatically invokes a macro (%INDXPAGE) to build an index Web page. We can write this or any other of our Web pages to a local file, or we can make use of one of the new features of SAS software, the FTP file access method.

```
filename wwwout ftp "gw001.html"   cd="public_html/air/"
         host="www.uchatham.edu"   user="air98"    pass="air98";
```

This method allows you to write files on any active machine on the Internet on which you have the proper authorization, by specifying the name of the machine, the userid, the password, the pathname, and the filename. The FTP access method is especially useful if your Web server is a public machine, while data processing is done on another machine behind a firewall. With this feature, you do not need to FTP manually each file to the Web server after generating it on the data processing machine.

The Index Page Macro

The macro code for building the index Web page is, from the program's viewpoint, page simply writing out text strings. For the most part, those strings are either basic HTML tags or formatted values of the appropriate variables. This step writes out HTML header code and creates a table. To keep the table from being too long, it is created with two columns, one with the first four campuses, the second with the remaining three institutions. The columns contain links arranged in unnumbered lists of departments, nested inside lists of colleges, nested inside a list of institutions. The link names are built from a common prefix ("gw1"), and the index numbers are assigned in the previous DATA step. A department's number is composed of its institution's number (INSTNO),

its division's number (SCHLNO), and its own number (DEPTNO). Divisional references are formed in the same fashion, with the departmental portion set to "00", while institutions have "000" for division and department.

Notice that the only hardcoded references in this section are to the institutional numbers of the first and last UChatham campuses in each subtable. All other references are generated from the data itself.

```
*************************************;
* Section 3. Build an index Web page.   *;
*************************************;

%macro indxpage;

   * Identify where the Web page will be written.              *;

filename wwwout ftp "gw001.html"   cd="public_html/air/"
         host="www.uchatham.edu"   user="air98"    pass="air98";

data _null_;
set index end=last;
by instno schlno deptno;
file wwwout;

if _n_=1 then do;           * Write HTML tags at the beginning of the file. *;
put @1 "<html>"/
    @1 "<head>" /
    @1 "<title>UChatham Persistence Report, &yr1</title>" /
    @1 "</head>" /
    @1 '<body bgcolor="#FFFFFF">' /
    @1 "<h2 align=center>"
       "University of Chatham Freshman Persistence Rates, &yr1</h2>" /
    @1 '<table border="0">' /
    @1 '<tr><table border="0"><tr><ul>';
end;

     * Start a new column with the first and fifth campuses.           *;
if first.instno and instno in(1,5) then put @1 "<td valign=top>";

     * Write out the campus link and start a new list for each campus. *;
if first.instno then put
    @1 '<li> <a href="campus/gw1' instno 1. '000.html">'
        instid $instb. '</a><ul>';

     * Write out the division link and start a new list for each division. *;
if first.schlno then put
     @1 '<li> <a href="campus/gw1' instno 1. schlno 1. '00.html">'
       schlname '</a><ul>';

     * Write out a link for each department.                          *;
put
    @1 '<li> <a href="campus/gw1' instno 1. schlno 1. deptno z2. '.html">'
       deptname '</a>';

     * End the unnumbered list for each division and campus when    *;
     *     appropriate.                                             *;
if last.schlno then put @1 '</ul>';
if last.instno then put @1 '</ul>';

     * End the column with the fourth and seventh campuses.          *;
if last.instno and instno in(4,7) then put @1 '</td>';
```

```
     * Close out everything that is still open and put a footnote when *;
     * you reach the last observation.                                  *;
if last then
   put @1 '</ul></td></tr></table></td></tr></table>' /
       @1 "<p> <h3>"
          "University of Chatham Planning/Persist.GR001/&sysdate </h3>" /
;
run;
%mend;
run;
```

The PROC TABULATE Macro

Section 4 consists of a macro, %LISTING, that is a PROC TABULATE filled with macro variable references. This procedure needs to be able to accommodate different levels of analysis (department, division, and institution) as well as different types of output (WEB1, WEB2, and PRT1). By replacing many of the parameters of a standard PROC TABULATE with these references, we allow other parts of the program to modify it as needed.

The macro variable &WHERE subsets the data for each department, division, or campus, so that the output from the TABULATE procedure contains only the appropriate observations. Values for the variables &VAR1, &VAR2, and &FMT change depending on the level of analysis, but are constant within a given level. These variables have their values set as the macro %LOOP1 works its way through all the levels.

The type of output determines the values that &SEPVAR, &RTS, and &INDENT take on. The %TAB2HTM macro from SAS Institute expects TABULATE output to contain separator characters between cells and does not expect nested row variables to be indented. Consequently, such output requires more space to accommodate the row variables. The macro %WEB2 uses this HTML formatting tool and sets these values accordingly. To produce attractive PROC TABULATE hardcopy print-style output, you can turn off the separator characters, indent any nested row variables, and reduce the row table space. The %WEB1 and %PRT1 macros use this type of output and set values for the appropriate macro variables when invoked.

By using a single macro to generate the output, regardless of type or level, we make modifications simple. Necessary changes to the PROC TABULATE output are made in only one place, insuring that all forms of output will be compatible.

```
************************************************************************;
* Section 4.  PROC TABULATE macro that runs Web and printed output. WEB1 *;
* & WEB2 macros call this macro, and PRT1 executes it for hardcopy output.  *;
************************************************************************;

%macro listing;

title1
"Graduation, Retention, and Persistence Rates for #byval(type1) in "
"#byval(instid)";
title2   "Specified Number of Semesters After First Enrollment";

   *  WHERE subsets the data for each department, division, or campus.  *;
   * SEPVAR is blank when for WEB2, and set to NOSEPS for WEB1 and PRT1.  *;

proc tabulate data=one(where=(&where))
     order=formatted missing &sepvar;
by type1 instid;
class acadyr yr sem type3 type2 instid school major1;

   *  FMT takes on different values for dept, division, and campus.    *;

format type2 &fmt;
```

```
   *  SPAN is set at the beginning of the program.                    *;

var count sem01-sem&span
          sch01-sch&span
          dpt01-dpt&span;



* VAR1 takes on different values for dept, division, and campus.      *;
   * VAR2 refers to the different variable sets for each level.         *;
   * VAR2 is DPT for departments, SCH for divisions, and SEM for campuses. *;
table
      &var1
      (type3=' ')*
      (acadyr=' '),
      (type2=' ')*
      (count=' '*sum='N'*f=comma6.
       &var2.01=' '*pctsum<count>='1'*f=5.1
       &var2.02=' '*pctsum<count>='2'*f=5.1
       &var2.03=' '*pctsum<count>='3'*f=5.1
       &var2.04=' '*pctsum<count>='4'*f=5.1
       &var2.05=' '*pctsum<count>='5'*f=5.1
       &var2.06=' '*pctsum<count>='6'*f=5.1
       &var2.07=' '*pctsum<count>='7'*f=5.1
       &var2.08=' '*pctsum<count>='8'*f=5.1
       &var2.09=' '*pctsum<count>='9'*f=5.1
       &var2.10=' '*pctsum<count>='10'*f=5.1)
       /rts=&rts.    &indent;
  *  RTS and INDENT vary based on the type of output desired.          *;
footnote1
"_____________________________________________________ &blanks ";
footnote2
"University of Chatham Planning/Persist.GR001/&SYSDATE &blanks ";
run;

%mend;
run;
```

The Hardcopy Macro

The macro %PRT1 is used to produce printed output. It sets the values for the macro variables &SEPVAR, &RTS, and &INDENT, and then invokes the macro %LISTING to produce the actual output.

```
*************************************************************************;
* Section 5.  PRT1 produces hardcopy output.                            *;
*************************************************************************;

%macro prt1;

   * Default macro values for PROC TABULATE in "%listing" *;

%let sepvar=noseps;
%let rts=20;
%let indent=indent=3;
run;

%listing;    * Produce the actual PROC TABULATE output.       *;
run;

%mend;
run;
```

The Plain HTML Macro

Our next macro, %WEB1, looks very similar to the beginning of %INDXPAGE; both use the FTP access method to designate the site of the output, and both use a DATA _NULL_ step to write HTML code to the specified output file. Where %INDXPAGE builds HTML tables, however, %WEB1 simply uses the HTML tag "<pre>" to tell browsers that what follows is preformatted text and should be displayed accordingly. Appropriate values for the variables needed in the macro %LISTING (&SEPVAR, &RTS, and &INDENT) are set by the CALL SYMPUT statements. Procedure output is redirected to the Web page file by PROC PRINTTO, and then %LISTING is invoked within this macro to produce the actual output from PROC TABULATE. A second PRINTTO closes the output file.

```
*************************************************************************;
* Section 6.  WEB1 uses PROC PRINTTO and PUT to redirect hardcopy as     *;
* preformatted text.                                                     *;
*************************************************************************;

%macro web1;

    * Identify where the Web page will be written.                *;

filename prt ftp "gw1&i.&j.&k..html"
         cd="public_html/air/campus/"
         host="www.uchatham.edu" user="air98" pass="air98";
run;

data _null_;
file prt;           * Write HTML tags at the beginning of the file. *;
put @1 "<html>"/
    @1 "<head>" /
    @1 "<title> UChatham Persistence Report, &yr1 </title>" /
    @1 "</head>" /
    @1 '<body bgcolor="#FFFFFF">' /
    @1 "<pre>";
    * All subsequent text will be preformatted.         *;
run;
    * Set values for TABULATE output.                   *;

%let sepvar=noseps;
%let rts=20;
%let indent=indent=3;
run;

proc printto file=prt;    * Route procedure output to Web page.     *;
run;

%listing;                 * Produce the actual PROC TABULATE output.     *;
run;

proc printto print=print; * Route procedure output back to default. *;
run;
%mend;
run;
```

The Fancy HTML Macro

While %WEB1 redirects the PROC TABULATE output to a Web page, it does nothing to take advantage of HTML's display capabilities. For more attractive Web pages, we use Web formatting macros within our macro %WEB2.

Like %WEB1, this macro uses the FTP access method to point the output to the desired location, and it uses CALL SYMPUT statements to set values for &SEPVAR, &RTS, and &INDENT. The %TAB2HTM macro (freely downloadable from SAS Institute's Web site) expects PROC TABULATE output to contain separator characters between cells, and it does not expect nested row variables to be indented. Consequently, such output requires more space to accommodate the row variables. Because this macro needs specific markers for cell boundaries in the output, the initial output from PROC TABULATE is not designed for regular printing. Horizontal and vertical lines are replaced by unprintable characters through the FORMCHAR option; indention of nested row variables is turned off and row separators are turned on; and the amount of space reserved for row titles is increased (because there is no indention allowed).

Once these changes have been set, %WEB2 calls %TAB2HTM once to begin capturing the output, then calls %LISTING to generate the output, and finally calls %TAB2HTM a second time to turn off the capture and begin the formatting. While this code has a spartan look to it, there is a good deal happening within the "black box" of the %TAB2HTM macro.

```
*************************************************************************;
* Section 7.  WEB2 uses SAS Web tools to build more extensive HTML code    *;
* than WEB1.                                                               *;
*************************************************************************;

%macro web2;

* Identify where the Web page will be written.                *;

filename prt ftp "gw1&i.&j.&k..html"
         cd="public_html/air/campus"
         host="www.uchatham.edu" user="air98" pass="air98";
run;

options nocenter ls=195 ps=85 formchar='828384858687888 98a8b8c'x;


    * Special FORMCHAR strings are required with TAB2HTM.     *;
    * See TAB2HTM documentation for operating system details. *;

    * Set parameters for PROC TABULATE output.                    *;

%let sepvar=;
%let rts=40;
%let indent=;
run;

%tab2htm(capture=on,runmode=b);    * Start capturing PROC TABULATE output.  *;
run;


%listing;         * Produces the actual PROC TABULATE output.      *;
run;

%tab2htm(capture=off,
         runmode=b,
         rlcolor=blue,
         clcolor=blue,
         bgtype=image,
         bg=../../marble1.jpg,
         encode=N,
         openmode=replace,
         htmlfref=prt,
         center=Y);
run;
```

```
     * Turns off the output capture and passes parameters for       *;
     * TAB2HTM processing.  See TAB2HTM documentation for details.  *;

%mend;
run;
```

The Big Loop Macro

Once the parts are built, we need one more macro to tie them together in a coherent whole; %LOOP1 fills that need. This coordinating macro consists of three loops -- the "i" loop for institutions, the "j" loop for divisions within each institution, and the "k" loop for each department within a school or college. Values of these index variables appear in the WHERE option in PROC TABULATE and in the name of the Web page where that output is directed. Because the values of the macro variables &WHERE, &VAR1, &VAR2, and &FMT are dependent on the level being displayed, CALL SYMPUT statements are used to set these variables at appropriate points within the macro.

Only the high value for the "i" loop (number of institutions) is hardcoded, because this is the value least likely to change. The maximum number of divisions changes from institution to institution, and this is reflected in the macro variable &MAXSCHL. Likewise, &MAXMAJ determines the high value of the "k" loop within a given division. Thus, if a new school or department appears in the data (or an old one disappears), the programmer does not need to modify the code; the changes are made automatically.

With the final step of invoking %LOOP1, we either generate the report in its traditional printed form or start the process of building hundreds of Web pages, as well as building a roadmap to allow users to find the exact page they need.

```
*********************************************;
* Section 8. LOOP1 runs all the other macros. *;
*********************************************;

%macro loop1;
                                  * Build index Web page when appropriate.  *;

%if "&type"="WEB1" %then %indxpage;
%if "&type"="WEB2" %then %indxpage;

%do i=1 %to 7;                             * Six campuses and UChatham total *;
   data _null_;
   set counter(where=(instno=&i));
   call symput("maxschl",compress(schlno));
   run;

   %do j=1 %to &maxschl;         * Number of divisions at given institution *;
       data _null_;
       set counter(where=(instno=&i and schlno=&j));
       call symput("maxmaj",compress(deptno));
       run;

       %do k1=1 %to &maxmaj;       * Number of departments within a division *;
           data _null_;
           call symput("k",compress(put(&k1,z2.)));
           run;

                * Macro variables for dept level Tabulate in "%listing" *;
           %let where=instno=&i and schlno=&j and deptno=&k;
           %let var1=(major1='Original Major: '),;
           %let var2=dpt;
           %let fmt=type2c.;
           run;
```

```
          %if "&type"="WEB1" %then  %web1;
              %else %if "&type"="WEB2" %then  %web2;
              %else  %prt1;

   * If the "%else" is removed, "%prt1" executes as well as the specified  *;
   * Web building macro.                                                   *;

          %end;                    * End departmental listings ("k loop") *;
          run;

     * Macro variables for division level Tabulate in "%listing" *;
   %let k=00;
   %let where=instno=&i and schlno=&j;
   %let var1=(school='Original Division: '),;
   %let var2=sch;
   %let fmt=type2b.;
   run;

   %if "&type"="WEB1" %then  %web1;
       %else %if "&type"="WEB2" %then  %web2;
       %else  %prt1;

   * If the "%else" is removed, "%prt1" executes as well as the specified  *;
   * Web building macro.                                                   *;

   %end;                              * End divisional listings ("j loop") *;
   run;

     * Macro variables for campus level Tabulate in "%listing" *;
%let j=0;
%let where=instno=&i;
%let var1=;
%let var2=sem;
%let fmt=type2a.;
run;

%if "&type"="WEB1" %then  %web1;
   %else %if "&type"="WEB2" %then  %web2;
   %else  %prt1;

   * If the "%else" is removed, "%prt1" executes as well as the specified  *;
   * Web building macro.                                                   *;

%end;                             * End institutional listings ("i loop") *;
run;

%mend;
run;

%loop1;                           * Kick it off.                          *;
run;
```

Conclusion

Using the FTP file access method and a series of macros, it is possible to build a program that can produce a multitude of Web pages with practically no human intervention required beyond submitting the program. By making the output dependent on macro variable references, the program can be run on different data sets at different times without extensive modifications.

This article illustrates the basic process for mass producing Web pages. Users are encouraged to adapt these ideas to their own needs, using different forms of output.

References

While this program makes use of the HTML formatting macro %TAB2HTM, there are other tools available from SAS Institute's Web site. Output from other procedures can be processed using the %OUT2HTM macro, while SAS data sets can be displayed with the macro %DS2HTM. You can download copies of all these macros at following URL:

http://www.sas.com/rnd/web/format/index.html

Questions and comments should be directed to:

Keith J. Brown
UNC-General Administration
Program Assessment & Public Service
P.O. Box 2688
Chapel Hill, NC 27515-2688
Telephone: (919) 962-1000
Fax: (919) 962-4316
Email: kjb@ga.unc.edu

Modified code is not supported by the author or SAS Institute Inc.