

Mass Producing Web Pages With SAS® Software

Keith J. Brown
University of North Carolina-General Administration
Chapel Hill, NC

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 features of 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 hard copy output when needed.

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. Most importantly, people who need information can obtain it directly from those who have it.

One method that SAS users have of making this information available is through SAS/Intrnet, which allows a

user with a thin client (a PC with just a browser) to use your web server to submit programs that will return HTML-formatted output to his computer. This product allows the user to access the most current information available.

Unfortunately, such convenience comes at a high price -- not only for the software, but also for the space needed to hold the data and the processing power needed to fulfill the data requests. While constantly-changing data might require such investments, many of us have data that is much more stable. 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 choose the specific web page that they need. In many cases, the total amount of space needed for the static pages will be a fraction of what would be needed for warehoused data, and processing power will be negligible.

Setting the Parameters

If you write your program so that the least technology-savvy person in your organization could with minimal guidance call up the program, modify it, submit it, and get the output, then you will have a program that you'll never have to worry about again.

In order to reach this nirvana of programming, you will need to make

use of the SYSPARM option and SAS macro variables (comments explaining what should go in the SYSPARM are also helpful). Anything in your program that should change when it is re-run--dates, types of output, or subsetting variables--should be specified as macro variables.

Our sample program takes in four pieces of information from the SYSPARM option--the current year, the current semester, number of semesters to be tracked, and the type of output desired. The accompanying DATA step creates 7 macro variables, some of them derived from the SYSPARM values, others created as constants. When this program needs to be re-run, the only line of code that should need modification is that one SYSPARM line.

```
*****;
* Current year, semester, span, *;
* and runtime (WEB1, WEB2, PRT1) *;
*****;
options sysparm="1998,1,20,WEB1"
        ovp nobyline ls=165 ps=55
        symbolgen nofmterr nosource2
        nocode nonumber ;

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

data _null_;
length blanks $ 100;
yr1=input(substr(sysparm(),1,4),4.);
sem=input(substr(sysparm(),6,1),1.);
spn=input(substr(sysparm(),8,2),2.);
type=upcase(substr(sysparm(),11));
blanks=repeat(' ',99);
if sem=4 then seq1=2*(yr1-1975);
    else if sem=1 then
        seq1=2*(yr1-1975)-1;
seq2=seq1-1;
seq3=seq1-spn;
call symput("yr1",compress(yr1));
call symput("seq1",compress(seq1));
call symput("seq2",compress(seq2));
call symput("seq3",compress(seq3));
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 SAS, some old (such as macros and PROC PRINTTO), and some new (such as the FTP file access method and SAS' web publishing tools). One of the first steps in adapting existing report-generating programs to produce web pages is to build "counter" variables to identify each desired report.

For our example, we'll use data from the fictitious University of Chatham. UChatham is made up of six constituent institutions; each institution has between four and six divisions, and each division has between one and fourteen departments. With pages for each department, division, institution, and the UChatham system, we will need to generate and index 231 web pages.

Since our index web page will contain links to reports down to the departmental level, the dataset 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 datasets 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.

Since SAS 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 dataset, we write out one observation per department; for the

macro counter dataset, we need one observation per division.

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

data one
  index(keep=instid instno schlno
        schlname deptno deptname)
  counter(keep=instno schlno
          deptno);
set one;
by instid schlname deptname;

retain instno 0 schlno 0 deptno 0;

if first.instid then do;
instno=instno+1; schlno=0; deptno=0;
end;

if first.schlname then do;
schlno=schlno+1; deptno=0; end;

if first.deptname then
deptno=deptno+1;

format instid $instb.;
length acadyr $9;
if sem=1 then
  acadyr=trim(put(yr-1,4.)) || "-"
           || trim(put(yr,4.));
if sem=4 then
  acadyr=trim(put(yr,4.)) || "-"
           || trim(put(yr+1,4.));

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

Building the Macros

With the index dataset created, the remainder of the program is a set of macros that can produce regular hard copy output (%PRT1), produce very basic web pages (%WEB1), or use SAS web publishing tools for more elaborate web output (%WEB2). All three options invoke a single macro (%LISTING) that produces the basic output.

While the sample program here produces only a single report, there is no 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 was compatible and controlled by the same parameters.

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, the FTP file access method.

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 webserver 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 webserver after generating it on the data processing machine.

The Index Page Macro

The macro code for building the index web page (shown in Section 5 of Appendix A) is, from SAS's viewpoint, simply writing out text strings. For the most part, those strings are basic HTML commands and formatted values of the appropriate variables. This step writes out HTML header code and creates a table composed of two sub-tables, or columns. 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 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 hard-coded references in this section are to the institutional numbers of the first and last UChatham campuses in each sub-table. All other references are generated from the data itself.

The PROC TABULATE Macro

Section 6 consists of two macros. %LISTING is invoked regardless of the type of output desired, while %PRT1 simply sets some default parameters before calling %LISTING. At its heart is a PROC TABULATE filled with macro variable references. The level being examined (department, division, or institution) determines the value of &WHERE, &VAR1, &VAR2, and &FMT, while the type of output determines the values that &SEPVAR, &RTS, and &INDENT take on. By using a single macro to generate all three types of output, we make modifications simple. Any necessary changes to the TABULATE output need be made in only one place, insuring that all forms of output will be compatible.

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 command "<pre>" to tell browsers that what follows is pre-formatted 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.

The Fancy HTML Macro

While %WEB1 redirects the TABULATE output to a web page, it does nothing to take advantage of HTML's display capabilities. For more attractive web pages, we use SAS's 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 uses CALL SYMPUT statements to set values for &SEPVAR, &RTS, and &INDENT. Because the %TAB2HTM 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; indentation is turned off and row separators are turned on; and the amount of space reserved for row titles is increased.

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 macro has a spartan look to it, much happens inside the %TAB2HTM "black box".

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. Since 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, since 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 form, or start the process of building hundreds of web pages, as well as a roadmap to allow users to find the exact page they need.

Summary

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. The SYSPARM option can be used to determine what sort of output is produced.

Appendix B shows a modification that will produce hardcopy and web output from the same submission. By eliminating the word "%else", we cause the %PRT1 macro to be invoked unconditionally. Regardless of the type of web output produced, we will always produce hardcopy output.

SAS Web Tools

You can download copies of the SAS web publishing macros at

www.sas.com

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Author Information

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

Appendix A

```

*****;
* Section 5. Build an index web page      *;
*****;

%macro indspage;

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;
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;
if first.instno and instno in(1,5) then put @1 "<td valign=top>";
if first.instno then put
  @1 "<li> <a href='campus/gw1' instno 1. '000.html'>"
    instid $instb. '</a><ul>';
if first.schlname then put
  @1 "<li> <a href='campus/gw1' instno 1. schlno 1. '00.html'>"
    school $school. '</a><ul>';
put
  @1 "<li> <a href='campus/gw1' instno 1. schlno 1. deptno z2. '.html'>"
    major1 $major. '</a>';
if last.schlno then put @1 "</ul>";
if last.instno then put @1 "</ul>";
if last.instno and instno in(4,7) then put @1 "</td>";
if last then
  put @1 "</ul></td></tr></table></td></tr></table>";
if last then put
  @1 "<p> <h3>"
    "University of Chatham Planning/Persist.GR001/&sysdate </h3>" /
;
run;
%mend;
run;

*****;
* Section 6. PROC TABULATE macro that runs web and printed output. WEB1      *;
* & WEB2 macros call this macro and LOOP1 executes it for hard copy output. *;
*****;

```

```

%macro listing;

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

proc tabulate data=one(where=(&where))
    order=formatted missing &sepvar;
by type1 instid;
class acadyr yr sem type3 type2 instid school major1;
format type2 &fmt;
var count sem01-sem&span sch01-sch&span dpt01-dpt&span;
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;
;
footnote1
"_____ &blanks ";
footnote2
"University of Chatham Planning/Persist.GR001/&SYSDATE &blanks ";
run;

%mend;
run;

%macro prt1;

data _null_;
    * Default macro values for Proc Tabulate in "%listing" *;
call symput("sepvar","noseps");
call symput("rts",compress("20"));
call symput("indent","indent=3");
run;

%listing;
run;

%mend;
run;

```

```

*****;
* Section 7. WEB1 uses PROC PRINTTO and PUT to redirect hard copy as *;
* preformatted text. *;
*****;

%macro web1;

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;
put @1 "<html>" /
    @1 "<head>" /
    @1 "<title> UChatham Persistence Report, &yrl </title>" /
    @1 "</head>" /
    @1 "<body bgcolor=\"#FFFFFF\">" /
    @1 "<pre>";
call symput("sepvar","noseps");
call symput("rts",compress("20"));
call symput("indent","indent=3");
run;

proc printto file=prt;
run;

%listing;
run;

proc printto print=print;
run;
%mend;
run;

*****;
* Section 8. WEB2 uses SAS web tools to build more extensive HTML code *;
* than WEB1. *;
*****;

%macro web2;

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='82838485868788898a8b8c'x;

data _null_;
call symput("sepvar"," ");
call symput("rts",compress("40"));
call symput("indent"," ");
run;

```

```

%tab2htm(capture=on,runmode=b);
run;

%listing;
run;

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

%amend;
run;

*****;
* Section 9. LOOP1 runs all the other macros.*;
*****;

%macro loop1;

data _null_;
if "&type"="WEB" then call execute('%indxpage');
run;

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

data _null_;
%if "&type"="WEB1" %then %web1;
  %else %if "&type"="WEB2" %then %web2;
  %else %prt1;
run;
%end;                                * End departmental listings ("k loop") *;

```

```

data _null_;
call symput('k','00');
call symput("where",trim("instno=&i and schlno=&j"));
call symput("var1",trim("(school='Original Division: '),"));
call symput("var2",trim("sch"));
call symput("fmt",trim("type2b."));
run;

data _null_;
%if "&type"="WEB1" %then %web1;
    %else %if "&type"="WEB2" %then %web2;
    %else %prt1;
run;
%end;                                * End divisional listings ("j loop") *;
run;

data _null_;
call symput('j','0');
call symput("where",trim("instno=&i"));
call symput("var1"," ");
call symput("var2",trim("sem"));
call symput("fmt",trim("type2a."));
run;

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

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

%mend;
run;

%loop1;
run;

```

Appendix B

```

data _null_;
%if "&type"="WEB1" %then %web1;
    %else %if "&type"="WEB2" %then %web2;
    %prt1;    * Note that the "%else" has been removed from this line;
              * %PRT1 will execute regardless of what value SYSPARM has;
run;

```