

Data-driven HTML

Keith Cranford, Marquee Associates, LLC
Mary Keinarth, Marquee Associates, LLC

Introduction

Internet and intranet applications are growing at a rapid pace. The SAS System is adapting to this environment with a new product, SAS/INTRNET, which allows dynamic data access based on user input. However, BASE SAS can also be used to produce dynamic Web pages in the sense that as data changes, pages can be updated periodically to reflect these changes. This paper will discuss programming techniques to produce such data-driven Web pages.

The main tool in SAS for generating these Web pages is the DATA step. With its powerful programming language, the DATA step provides the flexibility to dynamically produce HTML code. In conjunction with the DATA step, the SAS Macro facility provides a framework to standardize the code generation. Both of these tools will be addressed.

Data Step as HTML-generator

The DATA step can be used to generate HTML code dynamically. The flexibility of the DATA step programming language combined with the ability to include text and data from SAS data sets, makes it an excellent choice for building data-driven HTML code. Another useful tool is the Macro language. In this section some suggestions will be made on how to use these tools.

The following are some guidelines in using SAS to generate HTML code.

1. Start with a 'fixed' page, then make it data-driven.
2. Use macros for header and trailer code that is used for all pages.
3. Use data sets for data in tables.
4. Use data sets to create links.

These guidelines will be taken in turn and be used to introduce some HTML concepts and how to use SAS to generate the code.

The first suggestion is to start with fixed HTML source code. Any HTML development tool can be used for this task. Once this source code is

finalized, parts of the code which are fixed text versus the code which is data-driven can be identified. The fixed portion can then be handled in SAS with the PUT statement in a _NULL_ DATA step. Data sets, macros and macro variables can be used for the dynamic portion.

Since every page will have similar HTML code (except for changes in the heading) at the beginning and end, macros can be used including the page heading as the parameter for the header macro. The following are examples for header and trailer macros.

```
%macro header(title);

    put
    '<HTML>'                               /
    '<HEAD>'                               /
    '<TITLE>  &title  </TITLE>'           /
    '</HEAD>'                              /
    '<BODY  BGCOLOR="#FFFFFF">'          ;

    put
    '<TABLE cellpadding=5>'               /
    '<TR><TD><IMG SRC="logo.gif"           /
    '    width=90 height=90></TD>'        /
    '    <TD align=center><B><I>         /
    '        <FONT size=+3>'              /
    '        &title "                     /
    '        </FONT></I></B>'              /
    '    </TD></TR>'                      /
    '</TABLE>' ;

%mend header;

%macro trailer;

    put
    '</BODY>'                             /
    '</HTML>' ;

%mend trailer;
```

The macro HEADER puts the parameter TITLE as the banner heading on the page as well as in the heading in the body. In the latter case it is combined with a logo to give a consistent look to all pages. The macro TRAILER puts the closing tags for the <BODY> and <HTML> tags.

Data sets can be used to fill the data for HTML tables. Each observation will provide data for a row of the table. Beyond simply filling the table, however, the data step programming language can be used to change formatting based on the data values. The following example creates a page with a simple table.

```

data _null_;
  *--- Location of HTML file ;
  file htmlfile;
  *--- Header and ;
  *--- Table column headings ;
  if n=1 then do;
    %header(Department Summary);
    put
'<TABLE cellpadding=5 width=400>' /
'<TR><TD width=40%></TD>' /
'  <TD width=30% align=right>' /
'    <B>Oct 97</B></TD>' /
'  <TD width=30% align=right>' /
'    <B>Sep 97</B></TD>' /
'</TR>' ;
  end;
  *--- Read data ;
  set dept end=last;
  *--- Determine color of value ;
  if(oct97/sep97)>1.10 then
    color="#00FF00";
  else if (oct97/sep97)<0.90 then
    color="#FF0000";
  else color="#000000";
  *--- Row data ;
  put
'<TR><TD>' dept $dept. '</TD>' /
'  <TD align=right>' /
'    <FONT COLOR=' color '>' /
'      oct97 dollar10. '</TD>' /
'  <TD align=right>' /
'    sep97 dollar10. '</TD>' /
'<TR>';
  *--- Close table and ;
  *--- Trailer ;
  if last then do;
    put
'</TABLE>';
    %trailer;
  end;
run;

```

This table will contain one row for each department in the DEPT data set, giving the last two months sales. If sales have increased by more than 10% in the previous month, the sales figure appears in green; it appears in red if it decreased by more than 10%; otherwise, it appears in black. Of course, this program can be further generalized by determining the months to be included from the data set.

Also, the SAS macros could be used to standardize pages with similar tables. For example, supporting tables for the above table could be of the same basic form with just a change in the data set, category variable and HTML file to be created. Each of these could be parameterized into a SAS macro.

Finally, links can be incorporated into the tables as well. The most straightforward way to provide links is using the data values as the names of the linking tables. The above example code could be modified to provide a link to individual department pages. An additional assignment statement and a change in the row PUT statement will accomplish this.

```

link=trim(dept)||'.htm';
put
'<TR><TD><A HREF=' link '>' /
'  dept $dept. '</TD>' /
'  <TD align=right>' /
'    <FONT COLOR=' color '>' /
'      oct97 dollar10. '</TD>' /
'  <TD align=right>' /
'    sep97 dollar10. '</TD>' /
'<TR>';

```

Preparing SAS/GRAPH output

SAS Institute provides drivers for both GIF and JPEG graphics formats. PROC GREPLAY can be used to produce graphs in these formats. Specific examples will be provided at the conference.

Example Application

Consider a fictional computer retail store, Computer Town. This store has five departments: Accessories, Repair, Software, Systems and Training. Each of these departments sells various items. It would be beneficial for the owner to be able to see the last two months' sales for each department, highlighting good and bad sales figures. Details for each department could then be viewed by selecting a particular department, where sales for each item could be seen. Finally, selecting the monthly sales for an item would display detailed transactions. Examples of these tables are shown in the Appendix.

To accomplish this, the owner stores data as transactions, with a DATE, ITEM, CUSTOMER and AMOUNT. This data is summarized by (month of) DATE and DEPT or ITEM. The DEPT by DATE summary provides the top-level summary; the ITEM by DATE summary the second level. PROC MEANS is used for these summaries, with PROC TRANSPOSE used to reformat the data into the desired table form.

```

proc means data=retail.sales
  noprint;
  class dept desc date;
  format date monyy.;
  var amount;
  output out=new sum=;
run;

data s_all(drop=item)
  s_system
  s_access
  s_soft
  s_repair
  s_train ;
  set new;
  keep dept desc date amount;

  if _type_ in (1,5) then do;
    if _type_=1 then item='ZZZ';
    output s_all;
  end;

  if _type_ in (5,7) then do;
    if _type_=5 then desc='ZZZ';
    if dept='SYSTEM' then
      output s_system;
    if dept='ACCESSORY' then
      output s_access;
    if dept='SOFTWARE' then
      output s_soft;
    if dept='REPAIR' then
      output s_repair;
    if dept='TRAINING' then
      output s_train;
  end;

run;

%macro reform(dsn,var);
  proc sort data=&dsn;
    by &var;
  run;

  proc transpose data=&dsn
    out=retail.&dsn
      (rename=(col1=sep97
               col2=oct97)
       drop=_name_ _label_);
    by &var;
    var amount;
  run;
%mend reform;

%reform(s_all,dept)
%reform(s_system,item)
%reform(s_access,item)
%reform(s_soft,item)
%reform(s_repair,item)
%reform(s_train,item)

```

These summaries can then be displayed with SAS-generated HTML code based on the concepts discussed above. The following programs produce the three mentioned HTML pages.

```

proc format;
  value $dept
    'ACCESSORY'='Accessories'
    'SYSTEM'='System'
    'SOFTWARE'='Software'
    'REPAIR'='Repair'
    'TRAINING'='Training'
    'ZZZ'='Total';
  value $system
    '486'='486 Clunker'
    'P133'='P133 Speedster'
    'P200'='P200 Screamer'
    'P225'='P225 Demon'
    'ZZZ'='Total';
run;

*---- Department Summary page ;
filename htmlfile
  'c:\projects\SCSUG\summary.htm';

data _null_;
  *--- Location of HTML file ;
  file htmlfile;
  *--- Header and ;
  *--- Table column headings ;
  if _n_=1 then do;
    %header(Department Summary);
    put
      '<TABLE cellpadding=5 width=400>' /
      '<TR><TD width=40%></TD>' /
      '  <TD width=30% align=right>' /
      '    <B>Oct 97</B></TD>' /
      '  <TD width=30% align=right>' /
      '    <B>Sep 97</B></TD>' /
      '</TR>' ;
  end;
  *--- Read data ;
  set retail.s_all end=last;
  *--- Determine color of value ;
  if (oct97/sep97)>1.10 then
    color='''#00FF00''';
  else if (oct97/sep97)<0.90 then
    color='''#FF0000''';
  else color='''#000000''';
  *--- Row data ;
  link=trim(dept)||'.htm';
  if dept ne 'ZZZ' then
    put
      '<TR><TD><A HREF=''' link '''>' /
      '  dept $dept. '</TD>' /
      '  <TD align=right>' /
      '    <FONT COLOR=' color '>' /
      '      oct97 dollar10. '</TD>' /

```

```

'      <TD align=right>'
        sep97 dollar10. '</TD>' /
'<TR>';
    else
        put
'<TR><TD>' dept $dept. '</TD>' /
'      <TD align=right>'
        <FONT COLOR=' color '>'
            oct97 dollar10. '</TD>' /
'      <TD align=right>'
        sep97 dollar10. '</TD>' /
'<TR>';
*--- Close table and      ;
*--- Trailer              ;
    if last then do;
        put
'</TABLE>';
        %trailer;
    end;
run;

*---- System Summary page ;
filename htmlfile
'c:\projects\SCSUG\system.htm';

data _null_;
*--- Location of HTML file ;
    file htmlfile;
*--- Header and          ;
*--- Table column headings ;
    if _n_=1 then do;
        %header(Systems Summary);
        put
'<TABLE cellpadding=5 width=400>' /
'<TR><TD width=40%></TD>' /
'      <TD width=30% align=right>'
        <B>Oct 97</B></TD>' /
'      <TD width=30% align=right>'
        <B>Sep 97</B></TD>' /
'</TR>' ;
    end;
*--- Read data ;
    set retail.s_system end=last;
*--- Determine color of value ;
    if (oct97/sep97)>1.10 then
        color="#00FF00";
    else if (oct97/sep97)<0.90 then
        color="#FF0000";
    else color="#000000";
*--- Row data ;
    link=trim(item)||'.htm';
    if desc ne 'ZZZ' then
        put
'<TR><TD><A HREF="" link "">'
        item $system. '</A></TD>' /
'      <TD align=right>'
        <FONT COLOR=' color '>'
            oct97 dollar10. '</TD>' /
'      <TD align=right>'
        sep97 dollar10. '</TD>' /

```

```

'<TR>';
    else
        put
'<TR><TD>' desc $system. '</TD>' /
'      <TD align=right>'
        <FONT COLOR=' color '>'
            oct97 dollar10. '</TD>' /
'      <TD align=right>'
        sep97 dollar10. '</TD>' /
'<TR>';
*--- Close table and Trailer ;
    if last then do;
        put
'</TABLE>';
        %trailer;
    end;
run;

*---- Detail page for P225 ;
filename htmlfile
'c:\projects\SCSUG\P225.htm';

data _null_;
*--- Location of HTML file ;
    file htmlfile;
*--- Header and          ;
*--- Table column headings ;
    if _n_=1 then do;
        %header(P225 Demon - Oct97);
        put
'<TABLE cellpadding=5 width=400>' /
'<TR>' /
'      <TD width=25% align=center>'
        <B>Date</B></TD>' /
'      <TD width=50% align=left>'
        <B>Customer</B></TD>' /
'      <TD width=25% align=center>'
        <B>Sales</B></TD>' /
'</TR>' ;
    end;
*--- Read data ;
    set retail.sales end=last;
    where item='P225' and
        put(date,monyy.)='OCT97';
*--- Row data ;
    put
'<TR><TD>' date mmdyy8. '</TD>' /
'      <TD align=left>'
        customer $20. '</TD>' /
'      <TD align=right>'
        amount dollar10.
'      </TD>' /
'<TR>';
*--- Close table and Trailer ;
    if last then do;
        put
'</TABLE>';
        %trailer;
    end;
run;

```

Conclusion

BASE SAS can be a valuable tool in developing Web pages that can be easily updated as new data becomes available. The strength and flexibility of the programming language in the DATA step is what makes this possible. Also, the SAS Institute-provided drivers for converting SAS/GRAPH output into GIF or JPEG format,

provide an easy method for adding data-based graphics to a Web page.

Appendix

Examples of the HTML pages generated in the example application above are given below.

Figure 1: Department Summary Page

 <i>Department Summary</i>		
	Oct 97	Sep 97
<u>Accessories</u>	\$3,500	\$3,000
<u>Repair</u>	\$10,000	\$9,500
<u>Software</u>	\$40,000	\$30,000
<u>System</u>	\$52,500	\$59,000
<u>Training</u>	\$12,000	\$11,000
Total	\$118,000	\$112,500

Figure 2: Systems Summary Page

Computer Town <i>Systems Summary</i>		
	Oct 97	Sep 97
<u>486 Chunker</u>	\$4,500	\$9,000
<u>P133 Speedster</u>	\$12,000	\$15,000
<u>P200 Screamer</u>	\$12,000	\$15,000
<u>P225 Demon</u>	\$24,000	\$20,000
Total	\$52,500	\$59,000

Figure 3: Partial Detail Page

Computer Town <i>P225 Demon - Oct97</i>		
Date	Customer	Sales
10/03/97	ABC Co.	\$4,000
10/10/97	XYZ Co.	\$4,000
10/12/97	Joe Blough	\$4,000