Paper 187

Using SAS® Software to Provide Dynamic Web-enabled Reporting in the UNIX® Environment

Kent Nicholas, MatchLogic Inc., Westminster, CO

ABSTRACT

The explosive growth of the internet has enabled data of just about any topic to be readily available to anyone at anytime. Integrating the information delivery capabilities of the internet with the excellent report functionality of SAS ®, provides a solid application platform for delivering information to support business decisions. In this paper, we introduce a report application - a basic frequency generator - that obtains user input to dynamically generate information.

INTRODUCTION

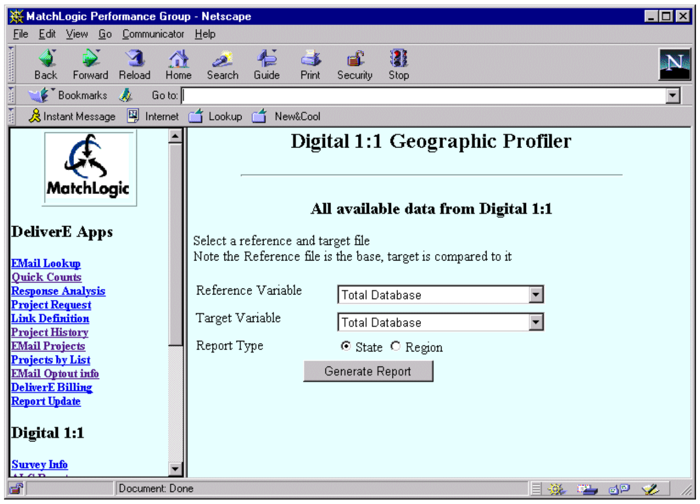
As an example of a dynamic web-enabled reporting application, this paper examines a frequency generation system. In this application, a user selects two variables which will be distributed across either states or census regions and compared to each other. The three main steps to generate dynamic web reporting are:

- Obtaining user requirements.
- Passing these requirements (parameters) to SAS.
- Using SAS routines to gather these requirements and generate the report.

OBTAINING USER REQUIREMENTS

This first step involves producing an initial HTML page to gather the report parameters. Building the HTML page is fairly basic and you will find a number of excellent HTML reference manuals on the market.

The HTML page consists of a form where, in this example, we gather the variables the user is interested in comparing and call the next program (see fig 1).



Digital 1:1 Geographic Profiler

All available data from Digital 1:1

Select a reference and target file
Note the Reference file is the base, target is compared to it

Reference Variable: Total Database
Target Variable: Total Database
Report Type: [✓ State] Region

Generate Report

Fig 1. Form to collect report parameters

Using the information collected on the form, the application executes a cgi script. This cgi script is the interface between the user and the SAS program. The following two lines of code executes the cgi script from with the html page.

```
<form method="POST" action="~/cgi-bin/geographic.cgi">
<input type=submit value="Generate Report">
```

PASSING PARAMETERS TO SAS

When the user presses the "Generate Report" button, the following geographic.cgi script is executed.

```
#!/usr/local/bin/perl
print "Content-Type:  text/html\n\n" ;
require('cgi-lib.pl') ;

# Gather info from html page ;
use CGI ;
$query = new CGI ;
$RefVarr = $query->param("RefVarr") ;
$RptType = $query->param("RptType") ;
$TarVarr = $query->param("TarVarr") ;
$PID = $$ ;

# Determine the variables to freq ;
open(VARFILE, "../sas/data/geographic.head") || die "can't open
variable list file: $!" ;
$VNum = 0 ;
$TNum = 0 ;
while ($inlist = <VARFILE>) {
   $VNum = $VNum + 1 ;
   $VText = substr($inlist,0,length($RefVarr)) ;
   $TText = substr($inlist,0,length($TarVarr)) ;
   if ($RefVarr eq $VText) {$RNum = $VNum;}
   if ($TarVarr eq $TText) {$TNum = $VNum;}
}
close(VARFILE) || die "couldn't close vartext.txt:  $!" ;

# run SAS program and generate html page ;
system `/opt/sas612/sas -sysparm
   "$PID*$RptType|$RNum-$TNum" geographic.sas -log /$PID.log` ;
open (HTMLFILE,"/opt/netscape/suitespot/sas/data/temp/$PID.html") ;
chmod 0777,"/opt/netscape/suitespot/sas/data/temp/$PID.html" ;
chmod 0777,"/opt/netscape/suitespot/sas/data/temp/$PID.gif" ;
while(<HTMLFILE>) {
      print $_ ;}
close HTMLFILE ;
unlink ("/var/tmp/$PID.log") ;
```

The first 8 lines of the script are primarily environmental and static. The next 3 lines assign the values from the html page into four variables: $RefVarr, $RptType, and $TarVarr, $PID - a useful variable that contains the UNIX process ID code. The next 12 lines of the script are used to determine which variables are being used. The file geographic.head is a flat file that contains the variables from the html page. This routine does nothing more than identify the numeric equivalent to the variable.

After establishing variables and assigning values to them, the script calls the SAS program geographic.sas using the following line of code.

```
system '/opt/sas612/sas -sysparm
   "$PID*$RptType|$RNum-$TNum" geographic.sas -log /$PID.log` ;
```

This system call passes the parameter string $PID*$RptType|$Rnum-$Tnum to the SAS program geographic.sas via the SAS macro variable &SysParm. The SAS log file is written out to $PID.log.

Once the SAS program completes, a print routine displays the results on the user's browser. Lastly, any logs and temporary files are cleaned up.
Hints:

- Use the unlink command to erase unneeded files.
- Change the mod of any permanent files so they can be taken care of later
- To avoid timing out the browser with long running processes, ask users for an email address to mail the report location when complete.

The following script works well for handling the problem of a users browser timing out on a long running process.

```
$PID = fork() ;
if ($PID == 0) {
   close (STDOUT) ;
   system (`/opt/sas612/sas -sysparm "$StrDate" //adhocyield.sas `) ;
   open Mail, "|mail $EMAdr" ;
   print Mail "Yield report for $StrDate - $EndDate complete and located at\n" ;
   print Mail "URL/reports/$ProcessID.html" ;
   close Mail ;
exit(0) ;
}
else {
print <<End_of_Notice ;
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY bgcolor=lightcyan>
<h4> yield reqport for $StrDate through $EndDate has been submitted you will
receive an email when complete</h4>
</body></html>
End_of_Notice
}
```

USING SAS TO BUILD THE DYNAMIC REPORT

Finally, some SAS programming! The following code receives the parameters from the cgi script and assigns them into SAS macro variables.

```
   %let L1 = %index(&SysParm,*) ;
   %let L2 = %index(&SysParm,|) ;
   %let L3 = %index(&SysParm,-) ;
   %let Len1 = %eval(&L3 - &L2 - 1) ;
   %let PID = %substr(&SysParm,1,(&L1-1)) ;
   %let RptType = %substr(&SysParm,(&L1+1),1) ;
   %let RefVar  = %substr(&SysParm,(&L2+1),&Len1) ;
   %let TarVar  = %substr(&SysParm,(&L3+1)) ;
```

The above code is used to determine the locations within the string &SysParm where the various parameters exist. Next, the program reads the flat file geographic.head and obtains the actual text values for report titles.

```
   filename  In3 "/opt/netscape/suitespot/sas/data/geographic.head" ;

   %macro Header ;
      %global RefName TarName ;
      data Header ;
         length Header $50 ;
         infile In3 delimiter='09'x missover dsd ;
         input Header ;
         if _N_ = &RefVar then call symput('RefName',trim(Header)) ;
         if _N_ = &TarVar then call symput('TarName',trim(Header)) ;
   %mend Header ;
```

This macro reads in the list of all variables then, if the pointer is equal to the specified reference or target parameter the title variable is assigned.

At this point the data can be aggregated a number of different ways: For example, use a proc summary, use a couple of frequencies, or build your own routine. In this example it was easiest to build a routine to access the data and generate percents.
The following code accesses a permanent SAS data set and calculates percents and an index.

```
   Data Agg ;
      Retain Tot1 Tot2 0 ;
      Set Data.Master ;
      If _N_ = 1 then do ;
         Tot1 = V&RefVar ;
         Tot2 = V&TarVar ;
      End ;
      Else do ;
         RefCnt = V&RefVar ;
         RefPct = RefCnt / Tot1 ;
         TarCnt = V&TarVar ;
         TarPct = TarCnt / Tot2 ;
         Indx = 100 * TarPct / RefPct ;
         if Index < 80 then Rank = 1 ;
         else if Index < 100 then Rank = 2 ;
         else if Index < 120 then Rank = 3 ;
         else Rank = 4 ;
      end ;
      Keep State RefCnt TarCnt RefPct TarPct Indx Rank ;
      Output ;
```

The data set Agg now contains 7 variables: State, the percent distributions and counts for both target and reference variables, the index (difference between the two percents), and Rank a grouping of the index.

The html report can now be generated. As you build more reports you may want to create specific macros to handle portions of the report building process. This is a good procedure to follow as all of your reports will begin to have the same look, and you don't have to keep referencing an html manual to remember the tag for some specific element. The following code outputs the html report from the above data set (Agg).

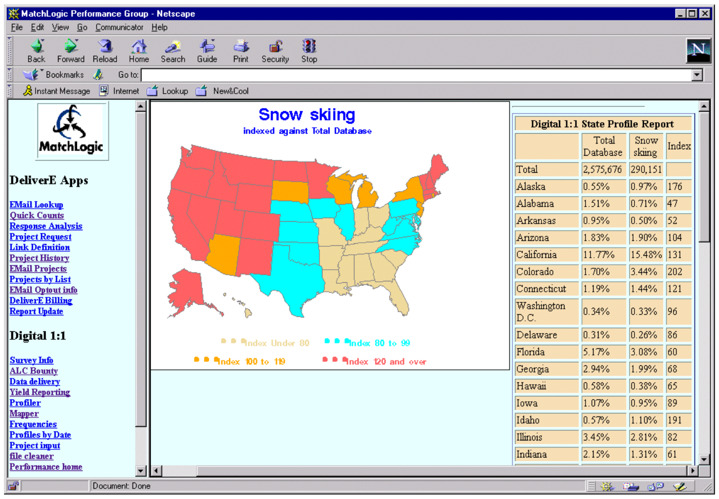
```
   data RptData ;
      retain TableCnt 0 ;
      file Out1 ;
      set Agg end=EOF ;
      if _N_ = 1 then
         put "<html>" /
             "<head>" /
             "<title>Digital 1:1 &RptTitle </title>"
             "</head>" /
             "<body bgcolor=lightcyan>" /
             "<img align=left    src= /temp/&PID..gif><hr>" /
             "<table border cellspacing=3 bgcolor=wheat
bordercolor=darkblue>" /
             "<th colspan=4 align=center> &RptTitle </th>" /
             "<tr align=center>"
             '<td>&nbsp;</td>' /
             "<td> &RefName </td>" /
             "<td> &TarName </td>" /
             "<td> Index    </td></tr>" /
             "<tr><td>Total</td>" /
             "<td>" RefCnt comma15. "</td>" /
             "<td>" TarCnt comma15. "</td>" /
             '<td> &nbsp;</td></tr>' ;
   else put
           "<tr><td>" State     "</td>" /
           "<td>" RefPct percent8.2 "</td>" /
           "<td>" TarPct percent8.2 "</td>" /
           "<td>" Indx  8.      "</td></tr>" ;
   if EOF then
      put  "</table><hr>" /
           "</body></html>" ;
```

Everything within the report is fairly easy to follow, the only tag that may be new is:
"<img align=left src= /temp/&PID..gif><hr>" /
This tag imbeds a graphic within the html document, in this case, a map produced by SAS/GRAPH®
The following code creates the map.

```
   goptions reset=all device=gif733 gsfname=Grf gsfmode=replace
HSize=6.0 VSize=4.0
         gunit=pct border ftext=swissb htitle=6 htext=3 ;
   options linesize=97 pagesize=55 date number pageno=1 ;
   title1 color=Blue "&TarName" ;
   title2 color=Blue "indexed against &RefName" ;
   pattern1 value=solid color=CREAM ;
   pattern2 value=solid color=CYAN ;
   pattern3 value=solid color=GOLD ;
   pattern4 value=solid color=ROSE ;
   footnote1 c=CREAM f=special h=6 f=swissb h=3 'Index Under 80'
          c=CYAN  f=special h=6 f=swissb h=3 'Index 80 to 99' ;
   footnote2 c=GOLD  f=special h=6 f=swissb h=3 'Index 100 to 119'
          c=ROSE  f=special h=6 f=swissb h=3 'Index 120 and over' ;

   proc gmap data = Agg
         map = maps.us ;
      id State ;
      choro Rank / levels=4 discrete nolegend coutline=gray ;
```

Fig 2 illustrates the final report that was created from the application using the Reference column of Total Database, and the Target column of interest in snow skiing.



Digital 1:1 State Profile Report

	Total Database	Snow skiing	Index
Total	2,575,676	290,151	
Alaska	0.55%	0.97%	176
Alabama	1.51%	0.71%	47
Arkansas	0.95%	0.50%	52
Arizona	1.83%	1.90%	104
California	11.77%	15.48%	131
Colorado	1.70%	3.44%	202
Connecticut	1.19%	1.44%	121
Washington D.C.	0.34%	0.33%	96
Delaware	0.31%	0.26%	86
Florida	5.17%	3.08%	60
Georgia	2.94%	1.99%	68
Hawaii	0.58%	0.38%	65
Iowa	1.07%	0.95%	89
Idaho	0.57%	1.10%	191
Illinois	3.45%	2.81%	82
Indiana	2.15%	1.31%	61

Fig 2 - final report

CONCLUSION

This basic example illustrates how to enable dynamic reporting over the internet with SAS. The information covered here can help you link the power of SAS and the convenience of internet reporting and publishing. With this solid platform, businesses can make data and information more quickly and easily accessible to support their business decisions.

REFERENCES

O'Reilly & Associates, 1996, *Programming Perl*
O'Reilly & Associates, 1997, *HTML The Definitive Guide*
SAS Institute Inc, 1993, *SAS Companion for UNIX Environments: Language, Version 6*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Kent Nicholas
MatchLogic Inc.
10333 Church Ranch Blvd.
Westminster, CO 80021
303.222.2000
Knicholas@MatchLogic.com