Dynamic Behavior from Static Web Applications
Ted Durie, SAS®, Overland Park, KS

ABSTRACT
Many Web applications, because of the infinite query combinations possible, require dynamic Web solutions. This typically means a programmer/administrator must configure, set-up, and license a Web and application server using SAS/IntrNet® software. If, by contrast, the possible query combinations are finite, limited, and manageable, then in some cases static Web tools and macros can be used instead. This approach requires minimal Web set up, and can attain the same functionality as dynamic solutions in some situations.

INTRODUCTION
Web Applications that require up to the minute information, or a large or infinite number selection combinations require dynamic web application. Examples of these might include eBay where from one user to the next you never know what they are interested in buying. As well, web applications that provided the current price of any stock listed on the NYSE or the current location of a Fed-Ex package could not use information from a static web system that was updated nightly. Moreover, a static web solution could be used to retrieve 401K information from a company like Boeing, GE, or Wal-Mart, but with over 100,000 employees in each, a dynamic approach is probably better. On average a static web report is about 5K, and with a 100,000 combinations possible, this application would require 500 Megs of storage.

As you consider whether an application should use a dynamic or static approach the key consideration is how many categories of information will this interface have. If a company wants to create weekly reports for all departments/divisions and that number is less than 500, then a static approach will work just fine. A web page that generates output based on the 435 Congressional Districts or on the roughly 250 US area codes could use this approach as well.

Implementing an application using this approach has several advantages. A dynamic solution requires a CGI Application Broker and set-up of an Application Server. Static only requires a web-enabled directory location. Application speed should improve as well. When a dynamic request is submitted ultimately in the background a SAS program is being executed. With static solutions the report has already been created, and displaying the information is simply a matter of accessing a directory location.

The solution presented in this paper uses concepts from several different SAS and PC disciplines. First a batch sas program that uses the AT Scheduler updates the application on a nightly basis. Second, the code incorporates the use of macros to generate detailed reports for all possible combinations of categories that exist in the data. Third the Output Delivery System (ODS) is used to convert information from PROC PRINT, REPORT, TABULATE into a static HTML table. Fourth when the application is interactively started an HTML screen appears that allows user to navigate to different parts of the interface.

The different pieces of the interface consist of US state maps broken down to a county level. A tabular HTML report that incorporates the use of traffic lighting and hyperlinks, and 3D pie chart created with the ‘ACTIVEX’ driver. All three of these top-level summaries will allow the user to drill-down to detailed reports. The county map and 3D pie chart have been hot spotted by DRILL= option.

NIGHTLY BATCH UPDATE
The code that creates and updates all the pieces of this WEB application are updated on a nightly basis by the AT scheduler on the PC.

Implementing an application using this approach has several advantages. A dynamic solution requires a CGI Application Broker and set-up of an Application Server. Static only requires a web-enabled directory location. Application speed should improve as well. When a dynamic request is submitted ultimately in the background a SAS program is being executed. With static solutions the report has already been created, and displaying the information is simply a matter of accessing a directory location.

The solution presented in this paper uses concepts from several different SAS and PC disciplines. First a batch sas program that uses the AT Scheduler updates the application on a nightly basis. Second, the code incorporates the use of macros to generate detailed reports for all possible combinations of categories that exist in the data. Third the Output Delivery System (ODS) is used to convert information from PROC PRINT, REPORT, TABULATE into a static HTML table. Fourth when the application is interactively started an HTML screen appears that allows user to navigate to different parts of the interface.

The different pieces of the interface consist of US state maps broken down to a county level. A tabular HTML report that incorporates the use of traffic lighting and hyperlinks, and 3D pie chart created with the ‘ACTIVEX’ driver. All three of these top-level summaries will allow the user to drill-down to detailed reports. The county map and 3D pie chart have been hot spotted by DRILL= option.

NIGHTLY BATCH UPDATE
The code that creates and updates all the pieces of this WEB application are updated on a nightly basis by the AT scheduler on the PC.
APPLICATION START-UP

The initial Application Start-Up menu allows the user to branch to Drill Down Map Application, The Sales Report that has used traffic lighting, or the Activex Drill Down Horizontal Bar Chart.

MAPPING APPLICATION

This section of the application retrieves a county map for any US state. The maps have been hot spotted using the DRILL= option. The next effect is that when you click on a given county a detailed tax report about the area appear.

TRAFFIC LIGHTING APPLICATION

When the user navigates to this portion of the application an HTML report with traffic lighting and hyperlinks appears. Selection of a currency figure in the right-most column takes you to a detailed report about that Country.

Expected Tax Revenue for KING county by city

<table>
<thead>
<tr>
<th>Name of City</th>
<th>Expected Tax Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algona</td>
<td>$4,554,719</td>
</tr>
<tr>
<td>Auburn</td>
<td>$99,002,533</td>
</tr>
<tr>
<td>Bellevue</td>
<td>$239,561,235</td>
</tr>
<tr>
<td>Black Diamond</td>
<td>$3,803,382</td>
</tr>
<tr>
<td>Bothell</td>
<td>$33,192,444</td>
</tr>
<tr>
<td>Butte</td>
<td>$67,457,693</td>
</tr>
<tr>
<td>Carnation</td>
<td>$3,342,099</td>
</tr>
<tr>
<td>Clyde Hill</td>
<td>$7,960,043</td>
</tr>
<tr>
<td>Des Moines</td>
<td>$45,469,421</td>
</tr>
<tr>
<td>Duvall</td>
<td>$7,447,798</td>
</tr>
<tr>
<td>Eastgate</td>
<td>$11,921,055</td>
</tr>
<tr>
<td>Enumclaw</td>
<td>$19,431,494</td>
</tr>
<tr>
<td>Fall City</td>
<td>$4,253,580</td>
</tr>
<tr>
<td>Federal Way</td>
<td>$181,634,850</td>
</tr>
</tbody>
</table>

Variance Summary by Country State and Product Type

<table>
<thead>
<tr>
<th>Country</th>
<th>State/Province</th>
<th>Product Type</th>
<th>Product</th>
<th>Amount of Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>U.S.A.</td>
<td>North Carolina</td>
<td>FURNITURE</td>
<td>BED</td>
<td>$35,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOFA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td>CHAIR</td>
<td>$5,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DESK</td>
<td>$80,000</td>
</tr>
<tr>
<td>U.S.A.</td>
<td>Texas</td>
<td>FURNITURE</td>
<td>BED</td>
<td>$15,500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOFA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td>CHAIR</td>
<td>$25,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DESK</td>
<td>$15,000</td>
</tr>
<tr>
<td>U.S.A.</td>
<td>Washington</td>
<td>FURNITURE</td>
<td>BED</td>
<td>$85,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOFA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td>CHAIR</td>
<td>$6,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DESK</td>
<td>$20,000</td>
</tr>
</tbody>
</table>
ACTIVEX BAR CHART

The third part of the application is a bar chart created using an activex driver. A chart created in this fashion allows the user to change the type of chart being displayed from a horizontal bar chart to a vertical or a pie chart. In addition the user can change the shape and colors of the bars/slices and text being shown out on the web. Additional functionality lets the user rotate the chart in three dimensions around the x, y, or z-axis. All of these changes to the appearance of the chart can be accomplished out on the web without having to execute any SAS code dynamically. The ability to change how the chart appears is built into the activex object.

### Analysis for country=U.S.A. and state=North Carolina and prodtype=FURNITURE and product=BED

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>YEAR</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BED</td>
<td>1995</td>
<td>$1,206.40</td>
<td>$1,539.60</td>
</tr>
<tr>
<td>BED</td>
<td>1996</td>
<td>$657.60</td>
<td>$952.80</td>
</tr>
<tr>
<td>BED</td>
<td>1997</td>
<td>$1,030.40</td>
<td>$568.00</td>
</tr>
<tr>
<td>BED</td>
<td>1998</td>
<td>$1,019.60</td>
<td>$957.60</td>
</tr>
<tr>
<td>BED</td>
<td>1999</td>
<td>$1,324.00</td>
<td>$720.00</td>
</tr>
<tr>
<td>BED</td>
<td>2000</td>
<td>$476.40</td>
<td>$1,126.40</td>
</tr>
<tr>
<td>BED</td>
<td>2001</td>
<td>$301.00</td>
<td>$1,529.60</td>
</tr>
<tr>
<td>BED</td>
<td>2002</td>
<td>$123.00</td>
<td>$967.60</td>
</tr>
<tr>
<td>BED</td>
<td>2003</td>
<td>$162.00</td>
<td>$709.00</td>
</tr>
<tr>
<td>BED</td>
<td>2004</td>
<td>$230.00</td>
<td>$1,101.00</td>
</tr>
</tbody>
</table>

Return to Previous
DRILL-DOWN PIE CHART

Most of the graphics routines inside SAS GRAPH allow users to drill down from one chart to some other kind of detail analysis. PROC GCHART using a PIE3D statement is no different. The last part of the application lets users select one of the slices of a 3D pie chart and thereby navigating to a detail report created by PROC TABULATE.

CRITICAL PROGRAMMING CONCEPT

Much of the functionality created in this interface was obtained by writing very little code at all. By simply setting DEVICE=ACTIVEVEX in a GOPTIONS statement a programmer can create an interactive interface that allows many of the appearance parameters to be changed on the fly. As well to convert output created by some procedure in the SAS system into HTML the Output Delivery System (ODS) was used. Successfully incorporating ODS into a program usually means the addition of only a few lines of code to your solution. To drill-down from a graph to a detail report requires more work as shown in the code section below in addition to setting HTML=DRILL in side the graphing procedure, one must also create the hyper-link one is navigating to inside the DATA step.

Data combine;
Length drill $60;
merge usc(rename=(cntyfips=county))
city(rename=(cntyfips=county))
cntyname(keep=county countynm);
by county;
drill='HREF=c:\workshop\winsas\graf8\[trim(countynm)]\.htm ;
total_city_tax=round(tot_county_tax*(pop/tot_county),.01);
format total_city_tax dollar15.;
run;

The hardest part of the application was creating traffic lighting inside the HTML report. Not only did the colors of the columns have to be assigned based off of some condition being true, but as well the web address of the hyperlink had to be constructed inside the data step as seen below.

data prd2;
set prd1;
Length tag $300 query_name $ 20 Query_type $ 130;
by country state prodtype product;
variance=actual-predict;
country1=country;
if country1='U.S.A.' then country1='USA';
if first.product then vartotal=0;
vartotal+variance;
If last.product then do;
query_type=trim(country)||"\","||trim(state)||"\","||trim(prodtype)||"\","||product;
query_name=substr(country1,1,3)||"_"||substr(state,1,3)||"_"||substr(prodtype,1,3);
if vartotal le -5001 then do;
tag="<a href=c:\training\web1\web1\[trim(query_name)]\.html" color="yellow">"||trim(left(put(vartotal,dollar12.))))"><font color="yellow"></font></a>";
end;
else if vartotal ge -5000 and vartotal le 5000 then do;
tag="<a href=c:\training\web1\web1\[trim(query_name)]\.html" color="orange">"||trim(left(put(vartotal,dollar12.))))"><font color="orange"></font></a>";
end;
else if vartotal ge 5001 then do;
tag="<a href=c:\training\web1\web1\[trim(query_name)]\.html" color="blue">"||trim(left(put(vartotal,dollar12.))))"><font color="blue"></font></a>";
end;
output;
end;
run;
To automate the creation of all the detailed reports macros were extensively used. In a dynamic web application to display a particular report a series of name value pairs would ultimately be passed back to sas program on an application server. The code would be executed for the combination of factors that the user was interested in. To obtain the same functionality in a static application all the potential selections has to be created before hand in batch sas program that is updated nightly.

MACROS were used to generate a separate static report for all the combinations possible as shown below.

```sas
%MACRO MAKE_REP;
data _null_; set prd2 end=last;
call symput('country'||trim(left(_n_)),trim(scan(query_type,1,'')));
call symput('state'||trim(left(_n_)),trim(scan(query_type,2,'')));
call symput('prodtype'||trim(left(_n_)),trim(scan(query_type,3,'')));
call symput('prod'||trim(left(_n_)),trim(query_name));
if last then call symput('totreps',trim(left(_n_)));
run;
ODS LISTING CLOSE;
%do R=1 %to &totreps;
ODS HTML Body="c:\training\web1\web1\&&file&R...html" style=BarrettsBlue;
  proc print data=prd1 noobs;
    where country="&&country&R" and 
      STATE="&&state&R" and
      prodtype="&&prodtype&R" and
      product="&&prod&R";
    title "Analysis for country=&&country&R and
      STATE=&&state&R and
      prodtype=&&prodtype&R and
      product=&&prod&R";
    footnote1 <a href="traflight.html">Return to
      Previous</a>:
    var product year actual predict;
run;
%end;
ODS HTML Body=c:\training\web1\web1\traflight.html
  style=brick;
title 'Variance Summary by Country State and Product Type';
  proc print data=prd2 noobs split='*';
    by country state;
    id country state;
    var prodtype product tag;
    label tag='Amount of Variance';
run;
ODS HTML CLOSE;
ODS LISTING;
%MEND;
```

In some case the number of potential selections were quite large. The mapping application had to create a detail report for every county in every state of the US. Approximately 5000 detail reports had to be updated nightly. An application of this size approaches the practical limit of a static interface. Still referencing a directory location should be faster than executes some section of SAS code on a remote server.

**CONCLUSION**

To implement a Web based application that has a large or unknown number of possible selections, a dynamic approach is required. If a user working with an interface needs the most current information a dynamic report is also needed.

Dynamic SAS Web applications are complex. They have to have a CGI Broker and Application Server running a SAS session in the background. Name value pairs must also be converted to MACRO variables. The user of the application must wait while a SAS program executes and generates the requested information. A static application simply requires that the reports be saved to a web enabled directory location on some server. No code is executed for the user to get the requested results. This should increase the speed of the application, and reduce the amount of time spent waiting for results to appear. Yes a JAVA or JAVA script program could be written to obtain the same functionality. The solution presented in this paper, however, with the exception of HTML, uses tools found inside the SAS System to generate results. This means another programming language does not have to be learned by existing SAS users. Granted, some of the macro and data step code is complex, but overall this presentation has positive attributes that should be explored by programmers.

**REFERENCES:**

SAS/Graph®, MACRO, Programming I and II, Web 1 and ODSGS training manuals concepts were referred to in this presentation.

**CONTACT INFORMATION:**

If you need more information or a complete copy of the material covered in this presentation then you can contact me at:

Ted Durie  
SAS Institute Inc.  
Corporate Woods, Building 40  
9401 Indian Creek Parkway  
Suite 600  
Overland Park, Kansas 66210  
Work Phone: 913-491-1166 ext 1311  
Fax: 913-491-4194  
Email: Ted.Durie@sas.com