

A "Real" Drill-Down Java Graph

Jenine Eason, AutoTrader.com, Atlanta GA

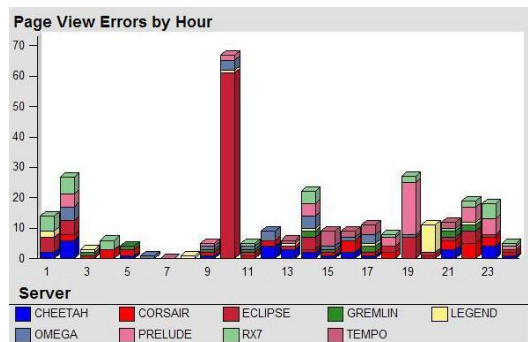
ABSTRACT

SAS/GRAPH® software enables you to summarize your data and present a summary using various charting options and view it from different directions. Drill down graphs let you slice and dice ONLY what you see in the main chart. To literally drill into a graph for additional information and charting, further coding is required, but easy to do. The method presented makes a chart vastly more dynamic. This particular example shows a further break down of website activity during a 24 hour period.

A JAVA APPLLET GRAPH

The code below creates a Java Applet that displays a vertical grouped bar chart. This example highlights the ability of the Java Applet to identify subsets of information along each response axis.

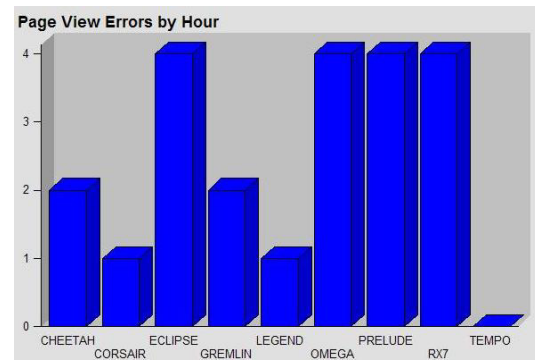
Java Applet Graph



In this example, website error activity is captured. The need here is to identify periods of high error activity and group

them in a way that is helpful in identifying the source of the error. By clicking on hour 14, the following view is presented. This chart identifies these errors within each hour of a single day and subset by web server.

Drill Down View of Hour 14



Main Graph Code

```
goptions reset=all ctext=black device=JAVA;  
  
legend1 frame cborder=black  
shape=bar(1,1)cells  
label=(justify=center position=(top center) )  
position=(bottom center);  
  
ods html file="C:\pg_errors2.html"  
parameters=("DRILLDOWNMODE"=  
"LOCAL")  
attributes=("codebase"="C:\Program Files\SAS  
Institute\Shared Files\applets")  
archive="graphapp.jar";  
  
proc gchart data=tmp.graphpaper;  
vbar3d hour / subgroup=server type=SUM  
sumvar=errCNT maxis=axis1 discrete  
frame cframe=white woutline=1  
coutline=black caxis=black  
legend=legend1 subgroup=server;  
title 'Page View Errors by Hour';  
run;  
quit;  
ods html close;
```

LIMITATIONS

While the code creating this Java Applet Graph has a lot of valuable information and versatility, you can only drill in for a closer view of something that is already available on the main graph. Hour 14 is a good example. It can be viewed on its own by drilling down into this bar of the chart. But the same information is already available on the main chart. By expanding and isolating the hour, no new information is obtained by drilling deeper into the chart.

ADDITIONAL GRAPHING NEEDS

There are other factors that can contribute to website errors. Specific URLs is one. Finding a way to incorporate URLs into a Java Graph would help isolate issues.

Adding another depth to the Proc GChart code above, *group=url*, created errors. The number of URLs that could be reported resulted in too many midpoints.

THE SOLUTION

The approach presented in this paper is to bring in another level of reporting to the Java Applet Graph. *The method to achieve this is to create a link from each bar in the chart to its own individual Java Applet Graph.*

To do this, a link needs to be created within the input data set. The *href* assigned to each hour will be the location of the additional Java Applet Graph.

```
%macro links;
data create_html_links; set tmp.graphpaper;
  length links $80.;
  %do I = 1 %to 24;
    if hour = &i then
```

```
links =href="file:///c:\hour||compress(&i.)|.html";
  /* if hour = 1 then
    links =href="file:///c:\hour1.html"; */
  %end;
run;
%mend links;
%links;
```

The original graph needs to be created as above, but this time, it acknowledges a drill target. It has a parameter *drilldownmode=local*. In this approach, the parameter *drilltarget=_self* is used.

Old

```
parameters = ("DRILLDOWNMODE"="LOCAL")
```

New

```
parameters = ("DRILLTARGET"="_SELF")
```

Now that the main graph is prepared to link to additional related graphs, each of the hourly graphs need to be created. A macro is used to replicate a graph for each hour of the day.

```
%macro hours;
%do i=1 %to 24;
  options reset=all ctext=black device=JAVA;
```

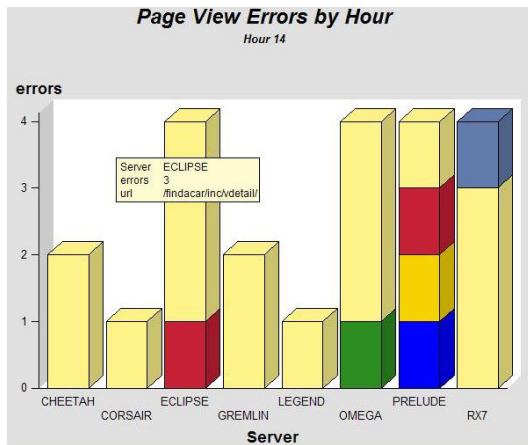
```
ods html file="C:\erhour&i..html"
  parameters=("drilldownmode"="local")
  attributes=("codebase"="C:\Program
  Files\SAS Institute\Shared Files\applets")
  archive="graphapp.jar";
```

```
proc gchart data=errors(where=(hour=&i));
  vbar3d server / levels=900
    type=sum sumvar=errcnt
    discrete frame cframe=white nolegend
    woutline=1 coutline=black caxis=black
    subgroup=url;
  title 'Page View Errors by Hour';
  title3 "Hour &i";
```

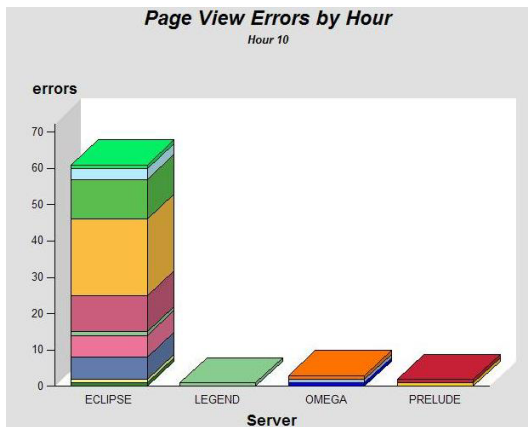
```
run;
ods html close;
%end;
%mend hours;
%hours;
```

Having the additional level of reporting, the following can be derived from the charts.

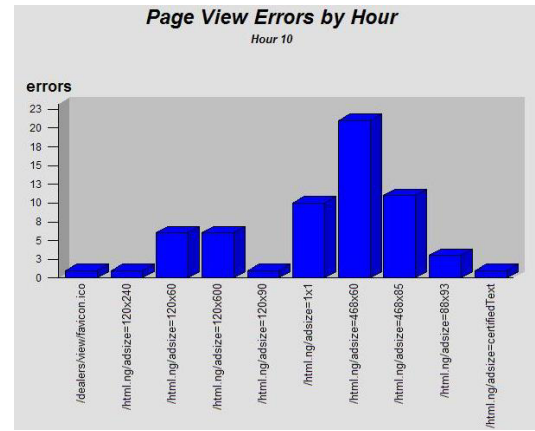
In hour 14, we can see that an individual URL is the culprit of the errors during this period. The majority of slices of each server in the graph below belong to one URL. This is valuable information that could only be identified with the additional reporting level.



In hour 10, we can easily see that it's a specific server with the errors rather than a URL.



Additionally, hour 10 can be further drilled to see the URLs comprising the errors that you see in the previous chart under one server.



CONCLUSION

Being able to bring in another level of reporting into an already dynamic application makes SAS/GRAPH a more valuable reporting tool. Links are easy to create and attach. Now you can really “drill-down” into a chart for more specialized information.

REFERENCES

SAS Inst. **Horizontal Grouped Bar Chart**, SAS/GRAPH software examples

SAS Inst. Customer Support team.

CONTACT INFORMATION

Jenine Eason
 AutoTrader.com, Business Intelligence
 5775 Peachtree Dunwoody Road
 Atlanta, Georgia 30342
 Email
Jenine.Eason@AutoTrader.com

SAS and all other SAS Institute Inc. product and service names are registered trademarks or trademarks of SAS Institute Inc.