Paper 29-27

Customizing the SAS/MDDB® Report Viewer

Phil Rhodes, Baylor University, Waco, TX

ABSTRACT

The SAS/MDDB® Report Viewer (MRV) included with SAS/IntrNet® is an effective way to deliver OLAP capability to user over a network. As an added bonus, SAS has provided ways to customize the appearance and functionality of the viewer. First, the appearance of many elements of the MRV can be controlled via the use of cascading style sheets (CSS). Second, there are a large number of global macro variables to change different aspects of the MRV; these can be set in the broker.cfg file or in a REQUEST INIT program. Since the MRV is implemented as an SCL class, method overrides can be used to replace SAS-supplied code with your own code for additional features. Finally, two of the programs in the MRV are also provided as examples in the Intrnet package; with a little work, these can be modified and replace the originals. This presentation will explore these options, starting with the basic MRV and ending with a very different customized version.

INTRODUCTION

In 1999, the Office of Institutional Research and Testing began the process of building a data warehouse. Almost immediately, it became apparent that the only feasible way to deliver access to the warehouse was over the web. Three potential OLAP tools for the warehouse were considered: the SAS webEIS® product, a custom-written front end using SAS/IntrNet, and the SAS/MDDB Report Viewer (MRV). Initial prototyping and user feedback indicated that the MRV was preferred over webEIS for ease of use, and would be quicker to implement than a custom-written front end.

The SAS/MDDB Report Viewer included with the SAS/Intrnet package is an effective thin-client OLAP solution. While the original interface is adequate, many organizations will likely want to modify the appearance and the functionality available to users. Fortunately, the MDDB Report Viewer (MRV) has many 'hooks' for customization. Using Cascading Style Sheets (CSS), you can change the appearance of most of the elements of the MRV. You can further modify appearance and some functionality using the many global macro variables listed the documentation (as well as some that are not listed). Finally, since the MRV is implemented as an SAS Component Language® class, you can override methods to provided additional features for your users.

CUSTOMIZING THE MDDB REPORT VIEWER

The three main methods of customizing the MRV are dealt with in individual sections, below.

CASCADING STYLE SHEETS

Nearly all of the HTML elements of the MDDB Report Viewer have class names, enabling the use of cascading style sheets (CSS) to modify the appearance of the MRV. A relatively complete and up-to-date list of these may be found online at the SAS web site (see References). Other class names may be found by examining HTML source from the MRV. To specify a CSS to use, add a hidden input field to your HTML:

```
<INPUT TYPE="hidden" NAME="CSS"
VALUE="http://servername/stylesheet.css>
```

or add the CSS parameter to the query string of report calls:

```
…&CSS=http%3A//servername/stylesheet.css
```

(note the URL encoding of the ':' – this may or may not be necessary depending on the browser, but should be done to adhere to standards).

The first change we made was to make all labels and header text boldfaced. The relevant CSS class names are *header*, *dimbox*, and *label*. Placing the lines

```
.header {text-decoration: underline;
            font-weight: bold}
.label {font-weight: bold}
.dimbox {font-weight: bold}
```

in a cascading style sheet definition will bold the headings and labels, as well as underline the header lines.

The second change was to adjust the alignment of the data in the report tables. By default, all data in all cells is centered. For uniformity, we wanted the labels left-justified and the data right-justified. Adding

```
.tdcell {text-align: right}
.trowcell {text-align: right}
.tcolcell {text-align: right}
.trowlab {text-align: left}
.rowlab {text-align: left}
```

to the CSS made these changes.

Additionally, we wanted to change the text for several of the labels. All of the text labels and error messages for the MRV are stored in the dataset SASHELP.WEBMSG. Using the FSEDIT procedure, we changed several messages to be easier to understand (such as replacing "Dimensions" with "Data Options" and "Table" with "Report Options". Note that this dataset is indexed and you may have to rebuild the index after editing. The SASHELP.WEBMSG dataset may be used by other applications in SAS/Intrnet and may also be overwritten by the installation of a new version of SAS; keeping a backup copy of both the original and the modified versions as well as a list of changes is highly recommended.

Two other changes were the rearrangement of the graph layout options and the addition of buttons to several screens. This was accomplished via SCL method overrides on the MRV WEBEIS class and is discussed below.

GLOBAL MACRO VARIABLES

SAS provides a large number of global macro variables that control various pieces of the reports generated. A relatively complete and up-to-date list of these can be found at the SAS website (see References). Using these macro variables, you can change the appearance and behavior of the MRV.

For example, to produce a 'canned' report that the user cannot modify, create a SAS program file that has the following statements:

```
%let _MRVANAL = NO;
%let _MRVSTAT = NO;
%let _MRNODIMBOXES = NO;
run;
```

and specify this as the REQUEST INIT program for your application dispatcher. Setting these three macro variables with any nonblank value will turn off the analysis variable list, the statistics list, and the Down/Across variable lists and the View Report button. Note that _MRNODIMBOXES was added in version 8.2 and will not work with older versions.

Other macro variables allow you to specify a site-specific help file (_MRVHELP), the location of the toolbar (_MRTBLOC), and the download to spreadsheet delimiter (_MRVSEP) for those locations that don't use a comma. The _MRVHELP parameter replaces the URL of the SAS help page with a URL you specify;

we provided a URL containing a basic tutorial on using the MRV.

SCL METHOD OVERRIDES

The most complex method of modifying the MRV is via SCL method overrides to modify the WEBEIS class that implements most of the functionality of the MRV. There are several stub methods provided by SAS that allow you to insert HTML at various points. Additionally, careful analysis of the flow of control and instance variables at the SAS website (see References) shows several other methods to override for additionally functionality.

Our first override was of the stub method **_outputHTMLAfterBody**. This method is called immediately after the <body> tag on the report page. We wrote a method that retrieved the label of the MDDB from the metadata and displayed it for the user (we felt it was important to keep the name of the data source in front of the user since many of our data sources have very similar sets of variables).

We created a new SCL class as a subclass of the main WEBEIS class and overrode the **_outputHTMLAfterBody** methods with the following code:

```
OPABODY: method / (signature='N');
  call send(_self_,'outputDataSrcTitle');
endmethod;
```

where **outputDataSrcTitle** is a new method we defined that retrieves the label of the MDDB from the metabase and outputs it as an HTML title:

```
DATASRC:
   method / (signature = "N");
   dcl num rc;
   dcl char(50) mddblabel mbname msg;
   dcl list mddblist metabase;

   metabase = getnitemn(envlist('L'),
            'METABASE',1,1,0);
   call send(metabase, '_get_mbname_',
           rc, msg, mbname);
   mddblist = makelist();
   call send(metabase,'_getDataAttr',
            rc,msg,mbname,mddb_,mddblist);
   mddblabel = getnitemc(mddblist,'LABEL');

   * Write html;
   rc=fput(htmlfile_,'<CENTER>');
   rc=fwrite(htmlfile_);
   rc=fput(htmlfile_,'<H3>Data Source:  '
            || mddblabel || '</H3>');
   rc=fwrite(htmlfile_);
   rc=fput(htmlfile_,'</CENTER>');
   rc=fwrite(htmlfile_);
   rc=dellist(metabase);
   rc=dellist(mddblist);
endmethod;
```

The second override was more complex. The MRV provides four toolbar buttons: Rotate, Download to Spreadsheet, Help, and Layout. We wanted to add buttons for a data dictionary and the data source selection page (so that the user did not have to 'back out' to that page if they had made many changes to a report). To accomplish this, we created two new methods, **outputDictionaryBtn** and **outputDataSourceBtn** to output HTML to display the button and the link to the correct pages. Then we overrode the **_outputToolbar** method of the MRV that actually outputs the toolbar to call the other toolbar buttons as well as our two new buttons. This method generates the <TR> and <TD> tags that cotain the toolbar buttons as well as the calls to the methods that generate the buttons themselves.

Here is a sample of the source code that adds the data source button to the toolbar:

```
rc=fput(htmlfile_,'<TD>');
rc=fwrite(htmlfile_);
call send(_self_,'outputDataSourceBtn');
rc=fput(htmlfile_,'</TD>');
rc=fwrite(htmlfile_);
if tbloc='3' or tbloc='4' then do;
  rc=fput(htmlfile_,'</TR>');
  rc=fwrite(htmlfile_);
  rc=fput(htmlfile_,'<TR>');
  rc=fwrite(htmlfile_);
end;
```

The method checks the toolbar location variable (*tbloc*) to determine whether to output a horizontal or vertical toolbar (for a vertical toolbar, a new table row is started after each button).

We also overrode the **_outputHdr** method to add help, dictionary, and data source buttons and the data source title. This method ouputs the initial HTML for the Report Layout page, including the content-type header, opening HTML tags, JavaScript variable functions, and the opening table tags. We rewrote this method and added a toolbar table with the desired buttons after the body tag:

```
/* Output toolbar buttons */
rc=fput(htmlfile_,'<TABLE><TR><TD>');
rc=fwrite(htmlfile_);
call send(_self_,'_OUTPUT_HELP_BUTTON_');
rc=fput(htmlfile_,'</TD><TD>');
rc=fwrite(htmlfile_);
call send(_self_,'outputDataSourceBtn');
rc=fput(htmlfile_,'</TD><TD>');
rc=fwrite(htmlfile_);
call send(_self_,'outputDictionaryBtn');
rc=fput(htmlfile_,'</TD><TD>');
rc=fwrite(htmlfile_);
rc=fput(htmlfile_,'</TD>');
rc=fwrite(htmlfile_);
rc=fput(htmlfile_,'</TR></TABLE>');
rc=fwrite(htmlfile_);
```

and the datasource title

```
/* Output data source title */
call send(_self_,'outputDataSrcTitle');
```

A third (and more complex) change we made was to rearrange the graph options on the Report Layout page. The current MRV has the Graph Source option first, the Graph Location option second, and the Graph Type option third. This was confusing to users, in that they saw that the Graph Source was selected, but the graph did not display unless a graph type was selected. We overrode the three graph layout option methods **_outputGraphLocOption**, **_outputGraphSourceOption**, and **_outputGraphList**, essentially exchanging their functions (this was easier than overriding the **_outputVariableSelForm** method). All three methods output static HTML and can be found in the MRV documentation. Thus, the overrides had the effect of making **_outputGraphSource** option output the Graph Type option, **_outputGraphLocOption** output the Graph Source options, and **_outputGraphList** output the Graph Location options.

CONCLUSION

While the MDDB Report Viewer included with SAS/Intrnet is usable 'out of the box', it is also easily customizable to suit your own needs. Careful selction of CSS parameters, macro variables, and method overrides can turn the MRV into an invaluable tool, customized to the needs of your users.

REFERENCES

Cascading Style Sheet tag list:
http://www.sas.com/rnd/web/intrnet/mddbapp/webcss.html

World Wide Web Consortium CSS Reference
http://www.w3.org/Style/CSS/
Global Macro Variables List
http://www.sas.com/rnd/web/intrnet/mddbapp/webvars.html
SCL Methods
http://www.sas.com/rnd/web/intrnet/mddbapp/methods/index.html

CONTACT INFORMATION

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

Phil Rhodes
Baylor University
Ste. 540 Robinson Tower
P.O. Box 97032
Waco, TX 76798
Work Phone: (254) 710-2061
Fax: (254) 710-2062
Email: Phillip_Rhodes@baylor.edu
Web:
http://bearhaus.baylor.edu/whdocs/DWPresentations.htm