Paper HOW-7

# Introduction to Interactive Drill Down Reports on the Web

Michael G. Sadof, Bedford NH
Louis T. Semidey, San Francisco CA

## ABSTRACT

Presenting data on the web can be an intimidating project for those who are not familiar with the technology. Luckily, SAS provides users with a method of presenting dynamic reports on a web server utilizing basic SAS syntax and even legacy reports. This workshop will walk you through several methods of utilizing Proc Report and will teach you some HTML code to present interactive reports on your web server. This technique requires you have several BI components running on your server but does not require advanced knowledge of SAS BI or HTML. In this workshop, we will start with the basics and develop a roadmap for producing dynamic reports with Stored Processes without using OLAP cubes or the OLAP Server. In the SAS BI environment, there are many ways of displaying data on the web. In this workshop, we will describe several of those techniques and create a project that utilizes them. The workshop will start by defining the necessary BI environment. We will continue by developing several simple reports at a detail and summary level, then enhancing with ODS and HTML to facilitate static drill downs and navigation. Then we will enhance the process by employing the SAS Stored Process Web Application to create dynamic reports. Finally we will introduce the use of AJAX (Asynchronous JavaScript and XML) for pages that do not require a full page refresh. This technique enables the development of rich internet applications. While the examples and code presented in the workshop will be simple the techniques can be leveraged and extrapolated to solve many real world presentation needs.

## BI Architecture

At the core of the SAS Intelligence Platform is the SAS Metadata Server. The metadata server basically controls all access to the metadata including users, libraries, reports and Stored Processes. Most of the metadata is stored in a set of SAS files known as the 'Foundation Repository'. The repository contains library definitions, report definitions, user logons, database passwords and much more. Our work here will be limited to identifying a library, defining a prompt and defining a Stored Process.
In the SAS Intelligence platform the engines that actually run SAS are called servers. There are many types of servers in BI one of which is the Stored Process Server which is designed to execute Stored Processes. A Stored Process is a SAS program that has been created in a certain way so that it can then be executed on the 'Stored Process Server' and send the results back to a web browser. We will be defining a stored process that 'streams' the data back to a web browser by utilizing the Output Delivery System (ODS). Figure 1 depicts the various layers in a typical BI installation. From the diagram it is seen that the middle tier or web server tier acts as a middle layer to communicate between the web browser and the server layer. In practicality once the layers are installed communication is handled by several very simple commands and the user only need know a simple set of syntax.

In order to run a report from a web browser the browser must communicate with the web server. In this case we will communicate with the web browser (middle tier) with the HTML or JavaScript language. We will formulate commands that can be interpreted properly by the Stored Process Server which in turn will process the SAS commands and subsequently deliver the report back to us in our web browser.

SAS has supplied several applications that run in a web browser. One of these applications is the Information Delivery Portal which can act as a portal for various types of web services. Another, which we shall utilize in this tutorial, is the 'SAS Stored Process Web Application'.

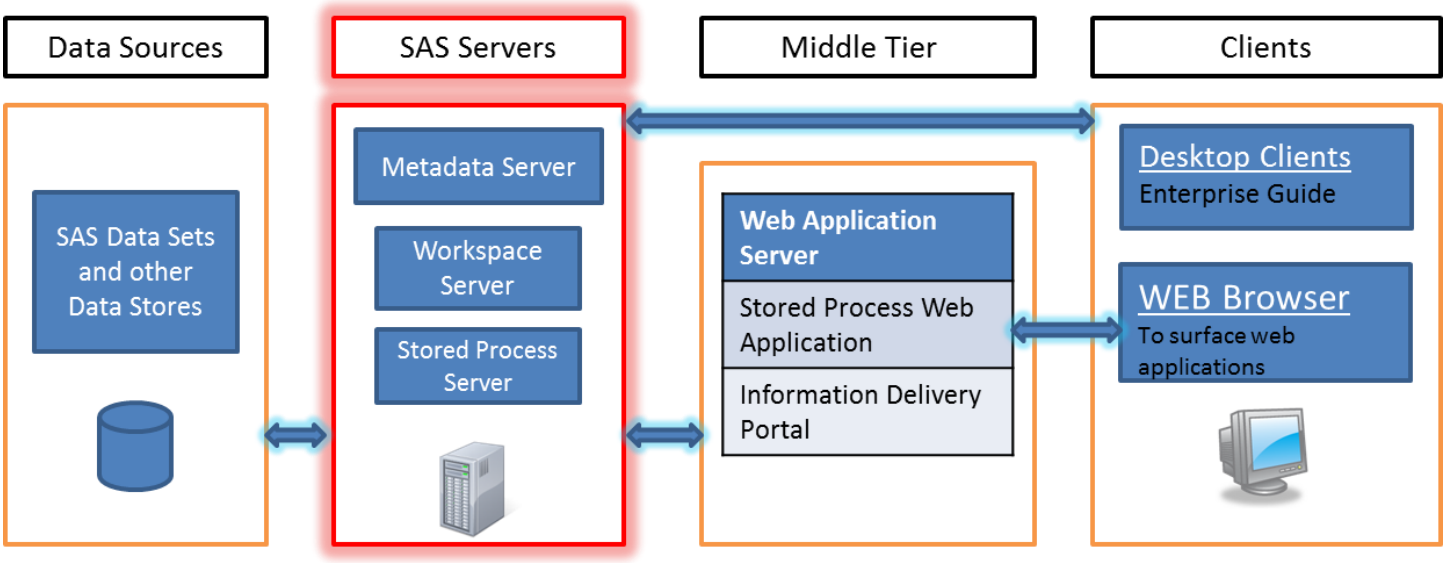



Figure 1. SAS BI Platform Architecture

## SAS Stored Process Web Application

The syntax (actually just the URL) to execute this application from your web browser will depend upon which Web Application Server is installed at your site. This could be WebSphere, WebLogic, or more commonly JBoss. These applications are installed at your site acting as a container and middle man to accept your commands and 'serve-up' your results--hence the name Web Server. We will only be exploring JBoss. To demonstrate this we will logon to our SAS installation and fire up a web server session.

Please note that the examples in this written version of the paper we conducted on my own server and the addresses and locations will be different than yours. Explanation of how to logon to this workshop's SAS installation can be found on a supplemental sheet.

## Web Server Demonstration

In this workshop we will be signing on directly to the SAS server. At your site you will most probably log in to the client and access your SAS server with an ip address or DNS name. If you are logged in directly to the server you can refer to it by the name 'localhost'.

1. Logon to your local workstation
2. Use Remote Desktop to logon to the SAS Server (see supplement)
3. Open a browser (Internet Explorer) window
4. Type this command in the address bar

   http://localhost:8080

   Note at your site you will most probably substitute your DNS name for localhost
   For instance at my site the SAS Server's DNS name is Dell-2850.mgs.local so I would logon as http://Dell-2850.mgs.local:8080

5. You will see a screen similar to this one indicating that JBoss is running. This is an easy way to determine if the WebServer is up and running.

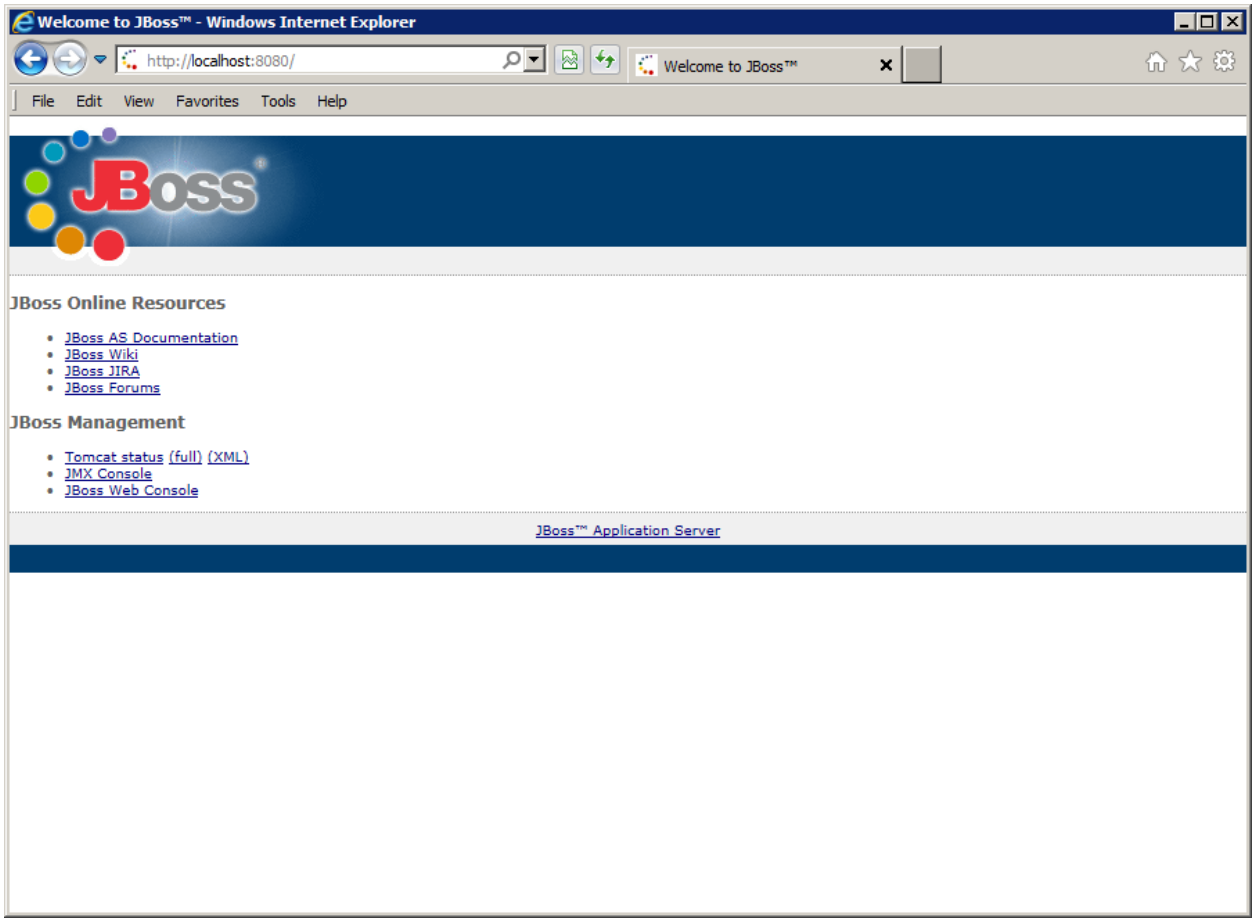



**Figure 2. JBoss Opening Screen**

6. Next type this command into the browsers address bar to logon to the SAS Web Server Stored Process Application

   http://localhost:8080/SASStoredProcess/do

7. This will bring up the familiar SAS LOGON Screen where you will enter your username and logon supplied on supplemental sheet.

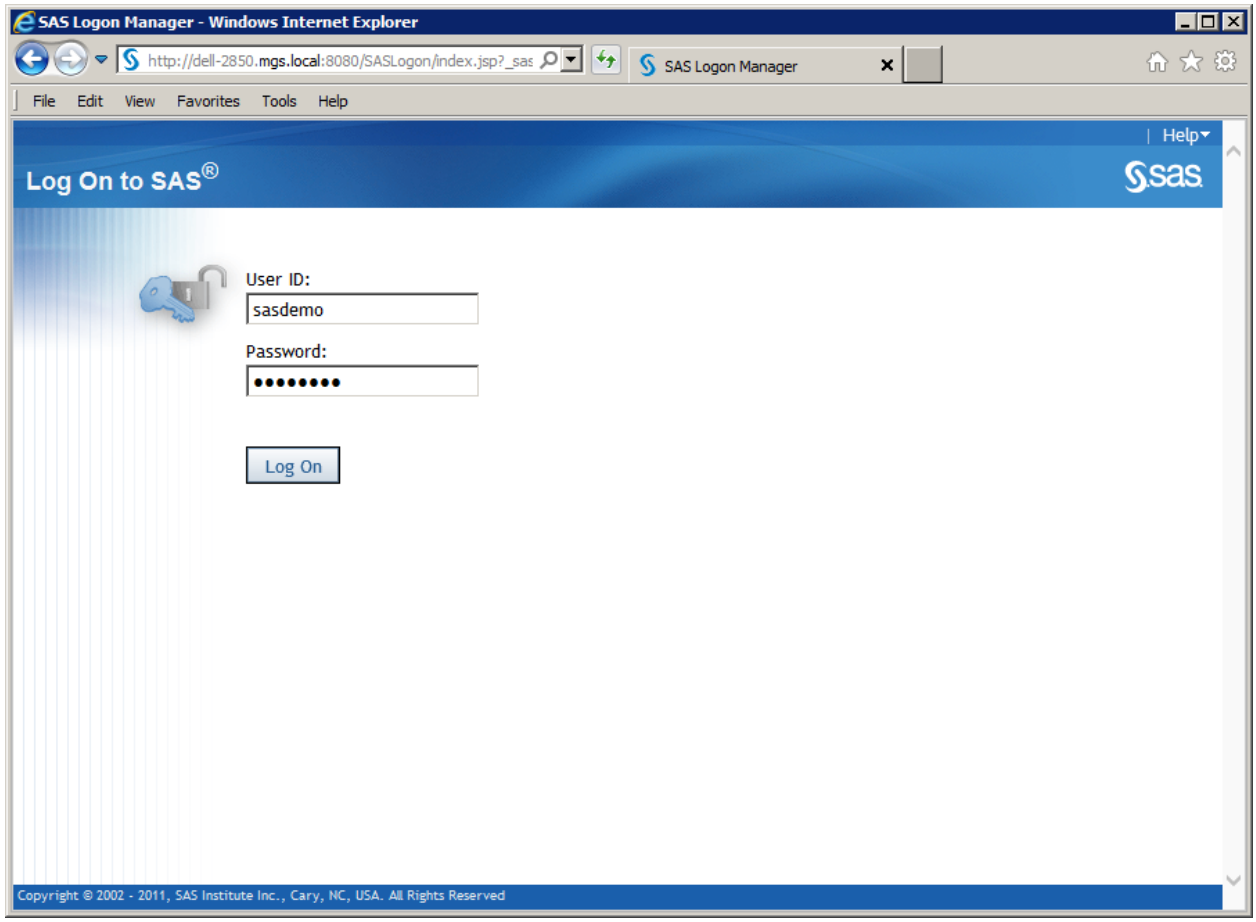



**Figure 3. SAS Logon Screen**

8. Next you will see the SAS Stored Process Web Application (Figure 3) where you may run a sample Stored Process. In our workshop we will be adding Stored Processes that can be surfaced through this application

9. Run the Sample: Shoe Sales by Region Stored Process by double clicking its name and explore by clicking some links. This will demonstrate how a SAS program in the background can be run from the web browser and can call other programs

10. Our next steps are to build a simple application like this and review ways to improve it.

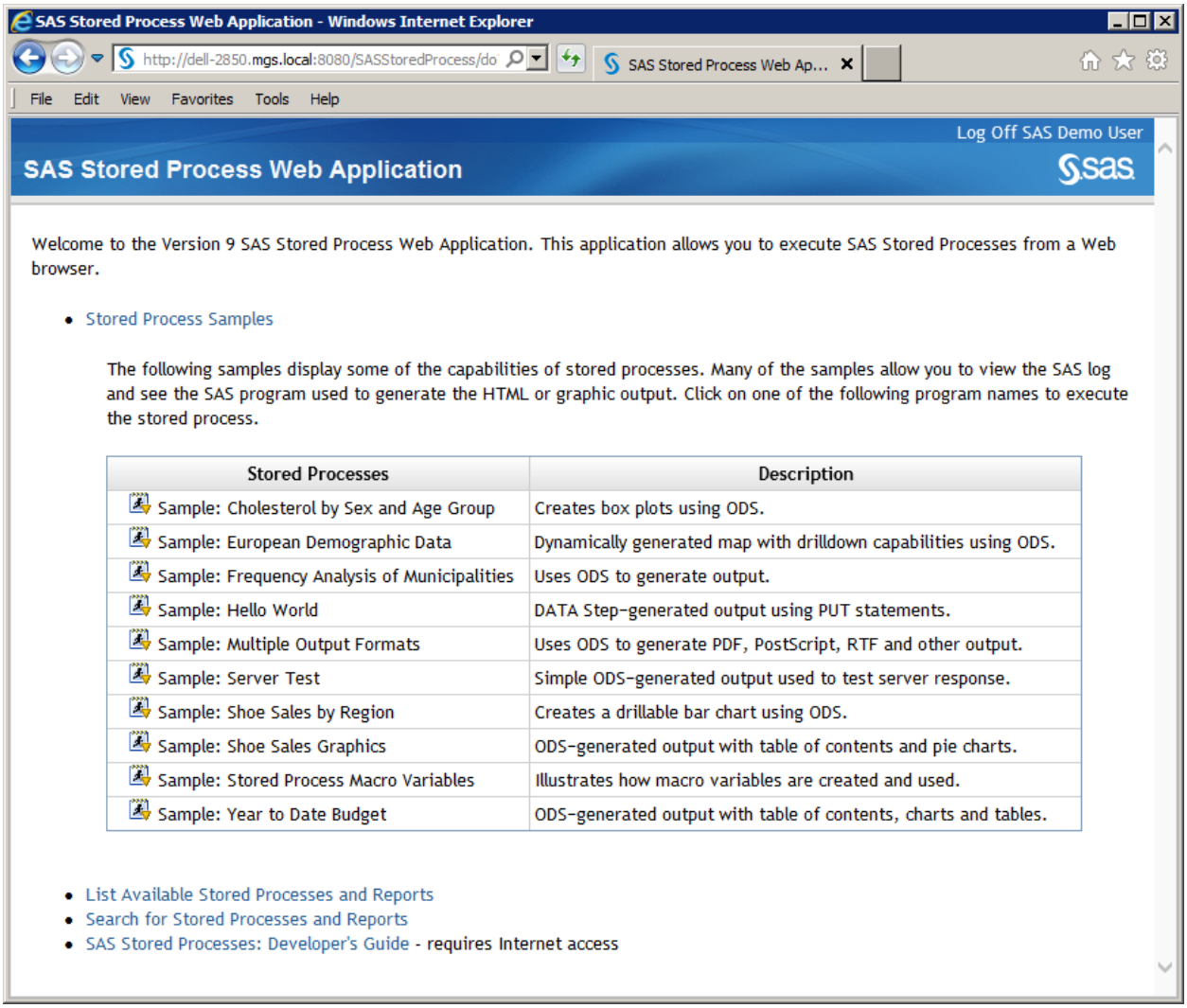



**Figure 4. SAS Stored Process Web Application Sample Reports**

## Simple Proc Reports for Demonstration

The technique we will be exploring today involves using ODS to create HTML pages that have embedded anchor tags which will use the HREF property to drill down to other reports.

- Proc Report
- ODS
- HTML Pages
- Anchor Tags
- HREF Property

## Example1 Demonstration of Proc Report

Example 1 creates several simple Proc Reports from the SASHELP data sets. It will get us familiar with the environment, setup and Proc Report.

1. Open up an Enterprise Guide Session
2. Open HOW7_Example1
3. Make certain that the output is set to HTML and SAS Report (Figure 4).
    a. Click on Toolbar Tools
    b. Click Options
    c. Click on Results General
    d. Make sure SAS Report and HTML are checked.
    e. Close window by clicking OK

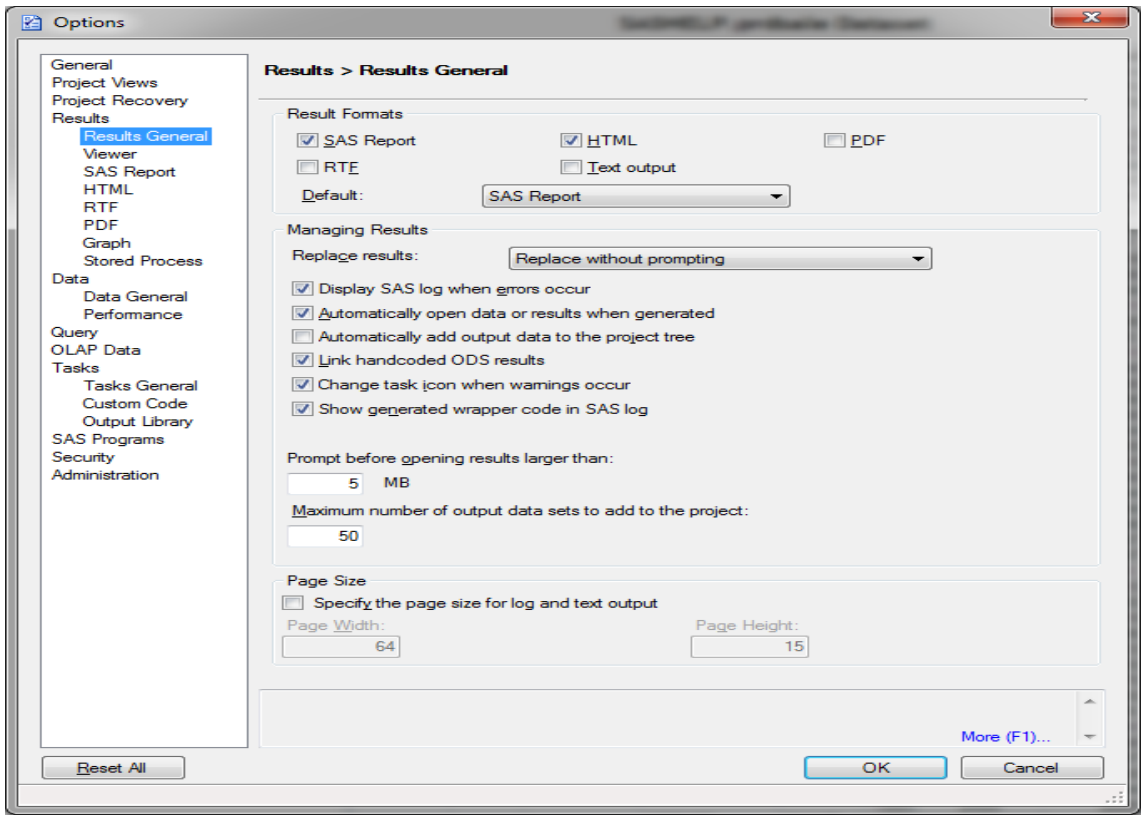



**Figure 5. EG Options**

4. Click on and open the first program example labeled Example1abcd.
5. Run each report separately and observe results
6. Highlight the first Proc Report labeled Example1a. This is a basic listing
    a. Right Click
    b. Run Selection on SASApp
7. Observe the results produced.
8. Highlight and run the second Proc Report (Example 1b) and run and notice how much more improved the results look with groups.
9. Observe that the use of an across variable will produce a much more readable but still voluminous result.
10. In the third example (Example 1c) a new computed column (COUNTRY_NAME) is added to enable the customization of sub headers. Observe also that a COMPUTE section was also added. We will use this type of compute section to create our drill down links in later examples.
11. The next report (Example 1d) will use the noprint option to eliminate duplicate columns and we use the computed column rather than the original country grouping. This is where we will eventually place our drill down links

Subsequent examples will be based upon the technique utilized in the Proc Report code from Example 1e.

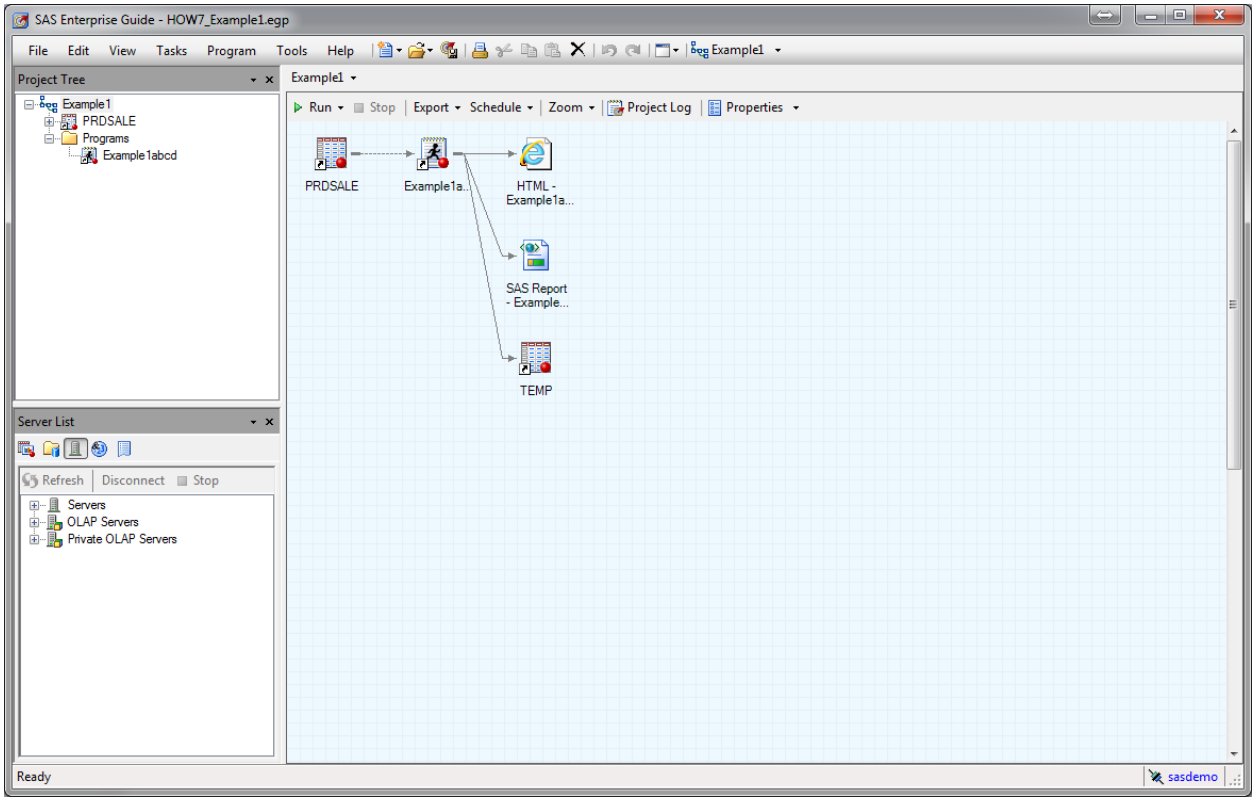



**Figure 6. Sample EG Screen for Example 1**

## Exercise: Build Example 2

Example 2 will build upon the basic Proc Report from Example1 which will:

- Create detail and summary reports
- Optionally add prompt to detail report
- Add ODS statements

1. Open the program (Example 1abcd).
2. Delete the reports 1a, 1b, and 1c so that all that is remaining is the last Proc Report labeled originally Example 1e
3. On the Project Tool Bar click save then choose 'Save As' and name it simply "Example 2a-Summary"
4. Save again by clicking File, Save As, then label it "Example 2b-Detail"
5. Delete Example 1abcd from project
6. Open Example 2a-Summary
7. Correct the Title Statement to read Example 2a.
8. The summary report (Example 2a) should only contain the following columns:

```
columns Country Country_Name Region Division Year , (Actual Predict)   ;
```

11. Save summary report, Open detail report
12. The detail report (Example 2b) should only contain the following columns:

```
columns Country Country_Name Region Division Prodtype product  Year , (Actual Predict)   ;
```

12. Delete the appropriate define statements as well.
13. You may refer to "Example 2ab_complete.sas" in your program directory
14. Run these jobs to view a detail and a summary report
15. Also note that this detail example runs a detail for every county. We need to put in a selection for an individual country.
16. The program Example 2a_Summary shows this syntax to select one country. Note: There is a copy of program in your directory if you need it.

```
Data = sashelp.prdsale (where= (Country = "&country." ));
```

17. To run this program we need to add a prompt for Country or use a statement like:

```
        %LET COUNTRY = xxxxxx;  ** where xxxxxx is country name **;
```

18. Next is to enhance these two programs (summary and detail) with ods statements. These ODS statements should be placed directly after the option statement. Use the appropriate **body=** clause for each program summary or detail.

```
%global odspath;
%let odspath=xxx;  * use valid path for your machine ;
ods listing close;
ods html path="&odspath."
     body="&country._Rep2_detail.html";  /*use for detail */
       /* body="Report2_Summary.html" ; */ /*use for summary*/
```

You will find the completed program in the project "HOW7_Example2.egp" and the programs "Example 2a – Summary.sas" and "Example 2b - Detail.sas". When you run the Detail program you will note that a prompt has been added to supply the selected country. We will not be using this prompting technique in our drill downs but we do need the macro variable "&country" the select statement shown to capture only one country in the detail report.

```
Proc Report Data = sashelp.prdsale(where= (Country="&country."));
```

We note that the output in HTML is closer to what we would like but there are no drill-down links as yet.

## Demonstration (please run)

Run Example2b – Detail report for each county in the sample dataset (CANADA, GERMANY, U.S.A.) This will create HTML files on the server that will be used in subsequent examples.

## Example 3 – Introduction of HTML Anchor Tags and HREF property

We will be surfacing the HTML version of these reports in our web browser. Happily SAS has built the HTML code for us. We just need to add a few tags which will create clickable links in our SAS-created HTML document.

The HTML Tag we will be using is the anchor tag which is surrounded by the **<A>** anchor open and **</A>** anchor close tags.

```
<A          > </A>
```

The HREF property will identify the URL. Insert the link to our HTML file within the quotes after HREF will enable the drill down. This needs to be in URL notation.

```
<A HREF=" " > </A>
```

The text before the anchor close tag </A> will be displayed on the HTML page.

```
<A HREF=" " >link text here without quotes</A>
```

We can then add the hint as a title property:

```
<A HREF=" " TITLE=" " > link text here without quotes</A>
```

That is the Anchor Tag in a nutshell, but obviously we need to a bit more work to get exactly what we want in the correct places and since we will be using macros the macro quoting functions come into play.

Open Project "HOW7_Example3.egp" and explore the Navigation Example. This example shows how to build clickable links within your title statement. Also introduced is the ODSESCAPE character that enables us to put SAS style specifications into the Title statement. It enables the use of various font sizes and colors and justification and most of the ODS style features. Run the "Example 3 – Navigation" program and observe that in the HTML output window you can see exactly what will be seen in your web browser. In the SAS Report window, which cannot interpret HTML code, you will see how the statement is built. This is not only instructive but very useful in debugging your code as the HREF's sometimes become very complicated. If you have run example2 correctly then these links should be clickable in the HTML version. The anchor tag should look like this where 'XX' is a valid location on your hard disk like **"c:\"** or **"c:\temp"**. **Report2_Summary.html** is the name of the html file for the summary report. Other links in the output are displayed with similar structure.

```
<A HREF="XX\Report2_Summary.html" Title="Hint with double quotes" >Displayed Link
Text</A>
```

At this point we have most of the building blocks needed.

- Summary and Detail Proc Reports
- Computed Variables to hold links
- Anchor Tags and Navigation Bars

When an HTML anchor tag is placed on the report and the report is rendered to HTML it becomes a clickable link as in the navigation bar.
All we need to do now is place the 'link' for the detail report into the compute block in the summary report so we can drill down to the detail from the summary.
The detail and summary reports for Example3 show the placement of the anchor tag and HREF in the compute block.

The compute block for example 2 looks like this

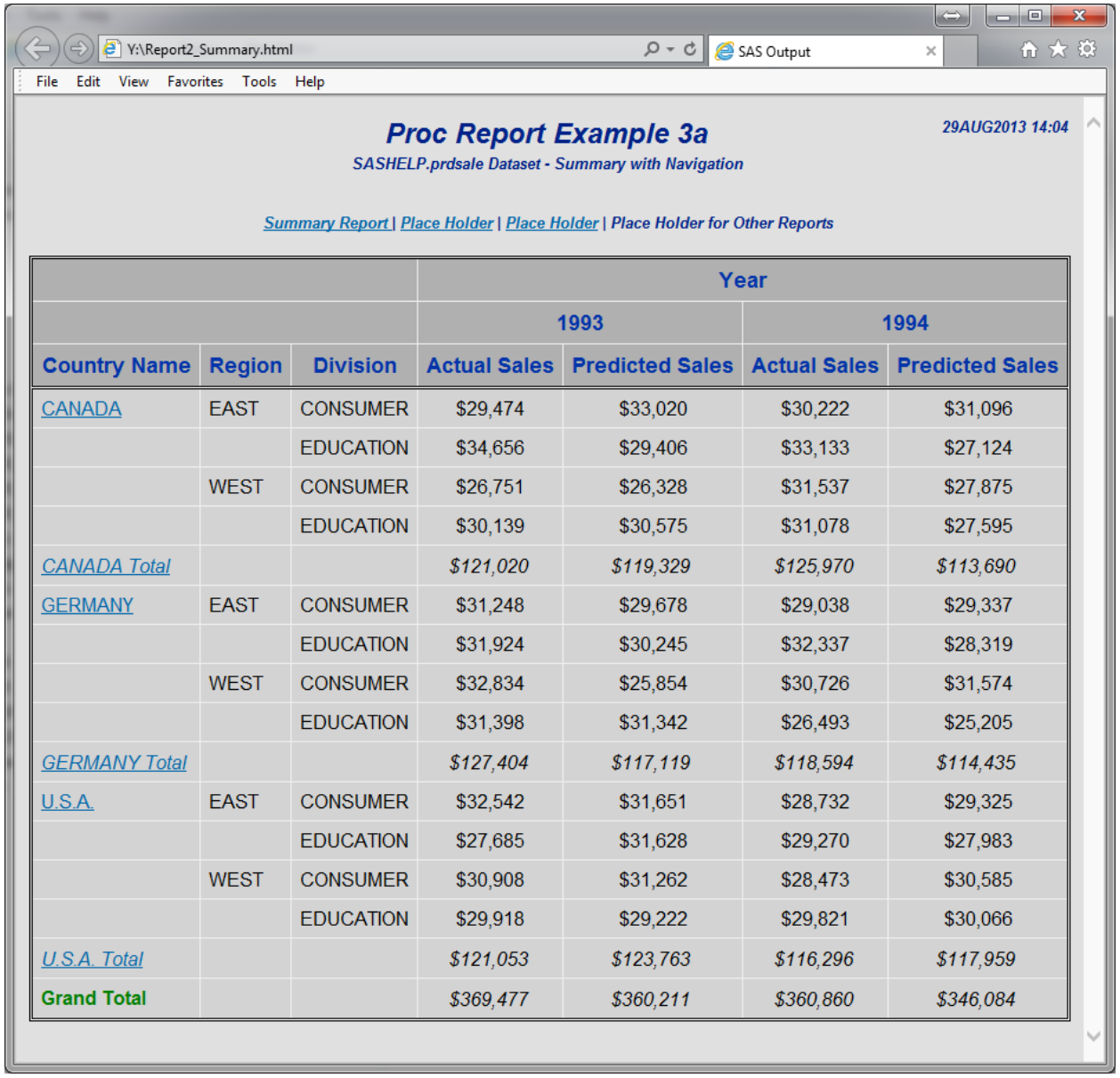
```
compute Country_Name / char Length=50;
Country_Name=country;
      if _BREAK_='COUNTRY' then country_name=trim(Country)||" Total";
            else if _BREAK_='_RBREAK_' then Country_Name='Grand Total';
endcomp;
```

A few minor changes to the compute block will place the achor tag and href directly on your report as the computed variable "Country_Name". I have also added a style statement to distinguish the "grand total" line in another color. You will notice the use of the concatenation symbols || to build the variable "Country_Name" as a string of characters with the appropriate quotes. Two quotes in a row **("")** has the effect of actually adding the quote to the string as we need for the HREF property of the tag. Spaces before or after the concatenation symbols || are ignored as are the line breaks. But be careful of spaces other places in the text Remember we want the name of the detail file to evaluate as `"Canada_Rep2_Detail.html"` so the trim(country) gives us the country name as retrieved from the group column.

```
compute Country_Name / char Length=150;
  X=COUNTRY; ** Hold Value of Country **;
  IF _BREAK_='COUNTRY' then x=trim(Country)||" Total";
  COUNTRY_name="<A href=""&odspath."
                  ||trim(Country)||"_Rep2_Detail.html""
                  Title=""This is the Hint"" >" ||
                trim(x) || "</A>";
  IF _BREAK_='_RBREAK_' then DO;
       Country_Name='Grand Total';
    call define(_col_,'style', 'style=[FONT_WEIGHT=BOLD foreground= Green ]');
    end;
endcomp;
```

## Demonstration

Open project HOW7_Example3.egp
Open Example 3 – Navigation
Examine the HREF links in the title statements.
Run program. Notice on the SAS Report output you can see the links fully formulated.
Notice on the HTML output you can see and operate drillable links.
Run the Summary Report and Click on a Country name to see detail
You will have to re-run program to get back to navigation or summary in as much as we have not yet fully implemented Navigation bar.




**Proc Report Example 3a**

29AUG2013 14:04

SASHELP.prdsale Dataset - Summary with Navigation

Summary Report | Place Holder | Place Holder | Place Holder for Other Reports

| | | | Year | | | |
|---|---|---|---|---|---|---|
| | | | 1993 | | 1994 | |
| Country Name | Region | Division | Actual Sales | Predicted Sales | Actual Sales | Predicted Sales |
| CANADA | EAST | CONSUMER | $29,474 | $33,020 | $30,222 | $31,096 |
| | | EDUCATION | $34,656 | $29,406 | $33,133 | $27,124 |
| | WEST | CONSUMER | $26,751 | $26,328 | $31,537 | $27,875 |
| | | EDUCATION | $30,139 | $30,575 | $31,078 | $27,595 |
| *CANADA Total* | | | *$121,020* | *$119,329* | *$125,970* | *$113,690* |
| GERMANY | EAST | CONSUMER | $31,248 | $29,678 | $29,038 | $29,337 |
| | | EDUCATION | $31,924 | $30,245 | $32,337 | $28,319 |
| | WEST | CONSUMER | $32,834 | $25,854 | $30,726 | $31,574 |
| | | EDUCATION | $31,398 | $31,342 | $26,493 | $25,205 |
| *GERMANY Total* | | | *$127,404* | *$117,119* | *$118,594* | *$114,435* |
| U.S.A. | EAST | CONSUMER | $32,542 | $31,651 | $28,732 | $29,325 |
| | | EDUCATION | $27,685 | $31,628 | $29,270 | $27,983 |
| | WEST | CONSUMER | $30,908 | $31,262 | $28,473 | $30,585 |
| | | EDUCATION | $29,918 | $29,222 | $29,821 | $30,066 |
| *U.S.A. Total* | | | *$121,053* | *$123,763* | *$116,296* | *$117,959* |
| **Grand Total** | | | *$369,477* | *$360,211* | *$360,860* | *$346,084* |

**Figure 7. Output from Example 3a**

Notice the date and timestamp in title. We will use this in future examples to verify that a dynamic Stored Process is being used rather than a static HTML page
In this example we have created series of static HTML pages with drill down capability.

## Example 4 – Build Stored Process

In Example 4 we will convert the static HTML reports to Stored Processes. Please note that they will still be static pages and will not get refreshed in every instance. The Stored Processes will be run from the SAS Stored Process Web Application that we examined earlier. First we will create a copy of Example3 and name it Example4

[HOW7_Example4.egp is same as HOW7_Example3.egp]
[HOW7_Example4a.egp is converted to point to Example4 files]
[[HOW7_Example4b.egp has completed changes]

1. Open Project “HOW7_Example4.egp” -- This is simply a copy of Example3
2. Open the program “Example 3a - Summary”
    a. Change Title Statement to read 4a
    b. ‘save as’ Example 4a - Summary
3. Open the program “Example 3b - Summary”
    a. Change Title Statement to read 4a
    b. ‘save as’ Example 4b - Detail
4. Open the program “Example 3a - Navigation”
    a. Change Title Statement to read 4
    b. ‘save as’ Example 4 - Navigation.
5. Save this project as HOW7_Example4a.egp – creating a working copy.
6. Open the program “Example 4 – Navigation”
7. Click on Create in the toolbar at top of program (not main toolbar). Shown in blue in Figure 8 below

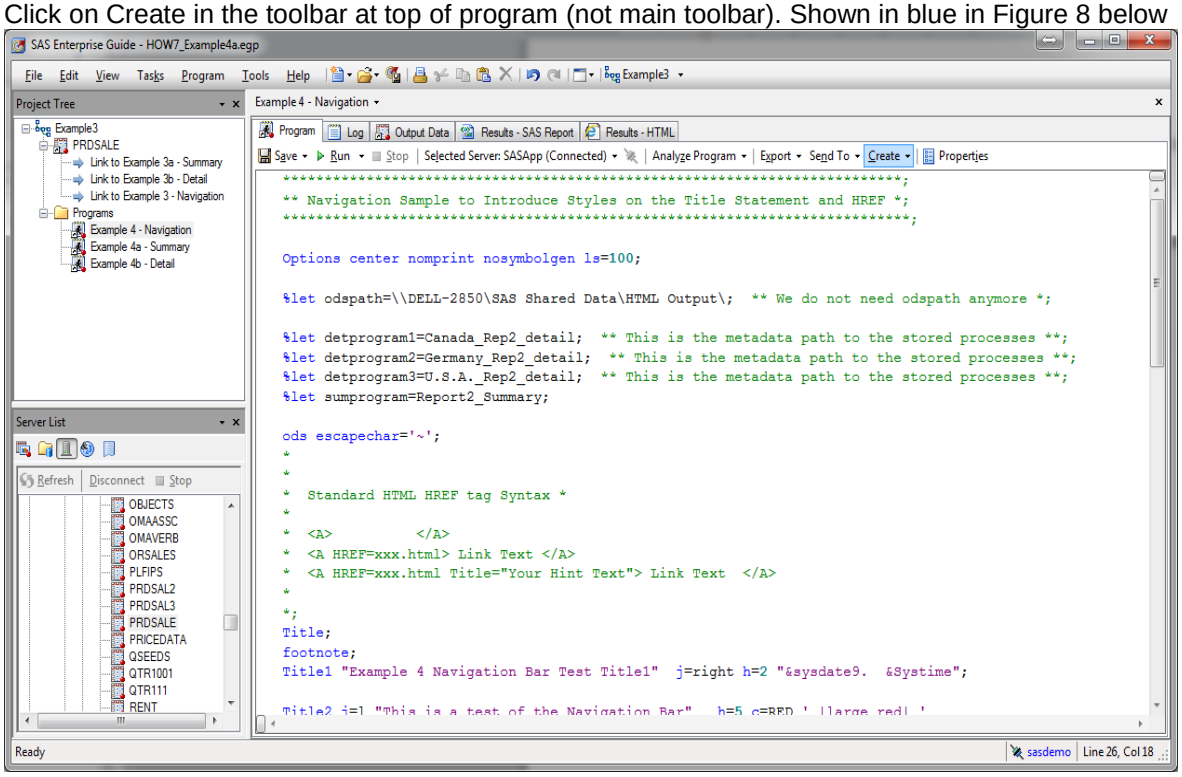



**Figure 8. Create Stored Process in EG**

8. Choose Stored Process
9. Enter “Example_4_Navigation” as the name for the Stored Process on the ‘Name and Description Screen’. Please note the underscores and lack of spaces in the name. Accept the other defaults and press NEXT
10. In the SAS Code Page – Comment out the lines for ODS if they exist (they exist on only in detail, and summary programs but not in the navigation code).

**Note: For the Summary and Detail programs you need to comment out the ODS Statements.** Stored Processes allocate a different type of file know as _webout which allows the steaming back to the browser. Look for lines like these and place and asterisk (*) in front as shown here:

```
*ods listing close;
*ods html path="&odspath" body="Report2_Summary.html" ;
*ods html close;
*ods listing;
```

11. On this screen “2 of 7” look for dropdown box “Include code for” and make sure all items (Stored Process Macros, Global Macro Variables, and Libname references) are checked.
12. Wait for the ‘3 of 7 Execution’ screen to fill in its defaults and check the ‘Overwrite Existing File’ box below the Source File Name box.
13. Enter “Example_4_Navigation.sas” as the source file name. Note again that underscores replace the spaces and dashes in the name. This will save a hard copy of the source file on your server in addition to the copy we have been working with in EG
14. Hit Next 4 times then Finish which will close the dialog
15. Do these for the other 2 programs in the project
16. Save the project, close and reopen it
17. Notice that the project now contains the original (non-Stored Process programs)
    a. Example – 4 Navigation
    b. Example – 4a Summary
    c. Example – 4b Detail

    And the Stored Process Metadata
    d. Example_4_Navigation
    e. Example_4a_Summary
    f. Example_4b_Detail

18. Save the project as HOW7_Example4b.egp for the next demonstration
19. Go to browser window and enter the following in the navigation bar as we did in an earlier example and we should see the new stored processes that were just built

    **http://localhost:8080/SASStoredProcess/do?**

20. Choose:

    List Available Stored Processes and Reports

21. Drill down to Shared Stored Processes and you will see:
    a. Example_4_Navigation.sas
    b. Example_4a_Summary.sas
    c. Example_4b_Detail.sas

22. Click on the Links run the process. The Detail report will not work yet but the summary report and the Navigation Report will drill down to our static HTML files Notice that these reports do not refresh the date in the header. These are static reports which must be run from EG. We would like these reports to be dynamically generated from the web page with live data. This will be accomplished in Example 5

## Example 5 – Add Stored Process Links to create dynamic report

The heart of the drill down call is the next HREF statement we are about to build.

**"http://localhost:8080/SASStoredProcess/guest?& program=/Projects/SESUG2013/Example 4a Summary&rep stat=Active&year=2013& debug=LOG"**

The breakdown of the format of this URL which will call the SAS Stored Process Web Application to run your stored process is as follows:

| | | |
|---|---|---|
| **http://** | Start of URL | |
| **localhost:8080/** | Server name and Port | localhost works for workshop since we are all signed on to our own virtual machine. You will have to use a valid DNS name or ip address |
| **SASStoredProcess/** | Name of SAS Application that will run our commands | Always remains same |
| **guest?** | Indicates guest user or regular logon which goes to the SAS logon manager.<br>? acts as a break between first part of URL and variable/value pairs that follow | We have set up the SAS Stored Process Web Application to accept guests without a password for demo purposes. See Appendix. |
| **do?** | | |
| **&_program** | Variable to store name of program we are going to execute | Ampersand (&) and underscore(_) are required |
| **Projects/SESUG2013/Example_4_Summary** | Metadata path and name of program | Spaces are allowed but you can use %20 to substitute for spaces<br>In this example the name of the metadata directory has spaces in it |
| **Your%20Name%20Here/Example_4_Summary** | No spaces in second example (if your metadata folder had spaces in the name) | |
| List of value pairs below | | |
| **&_program=/Projects/SESUG2013/Example_4a_Summary** (as shown above) | | |
| **&rep_stat=Active** | These variables are passed as macro variables to the stored process as if we had a statement like:<br>%let rep_stat=Active | |
| **&year=2013** | Passed to STP | |
| **&_debug=LOG** | Tells SAS to print LOG for debugging | |
| | | |

Note: The name of the program is the metadata name of the program and not necessarily the name of the program on the file system which in this case would be “example_4a_summary.sas”

## Exercise

We will start with the Stored Process Programs we built in Example4. In the interest of time I have prepared HOW7_Example5.egp, with the changes we need for each program
HOW7_Example5a.egp – Copy of Example4b
HOW7_Example5b.egp – Renamed modified programs
HOW7_Example5c.egp – Stored Processes

Example 5 Navigation
Example 5a Summary
Example 5b Detail

The main change is the replacement of the HREF statements to point to stored processes as described previously. Several macro statements have been added to point to the Stored Process metadata names rather than the static detail reports. All changes are noted below. In addition we will create stored processes from the code as we did in example 4.

### Example_5_Navigation

Change Title8 and Title9 to build new navigation bar.

```
Title8 j=l   h=4  '<A HREF="'
      "%SUPERQ(TSESSION)"
      '&_ACTION=EXECUTE'
      '&_program=' "&sumpgm"
      '&repstat='  "Active"
      '&year='     "2013"
      ' Title="You can place text here for a Hint when Hovering" '
                '>Link to Stored Process Summary </A>';

Title9 c=green j=l h=2
      '<A HREF="'
      "%SUPERQ(TSESSION)"
      '&_ACTION=EXECUTE'
      '&_program=' "&navpgm"
      '&repstat='  "Active"
      '&year='     "2013"
      '&_debug='   "LOG" '"'
       ' Title="You can place text here for a Hint when Hovering" '
           '>Link to Navigation Stored Process</A>';
```

Removed other **%LET** statements pointing to static HTML pages and added these **%LET** statements to point to stored processes rather than static html.

```
%let navpgm=/Projects/SESUG2013/Example_5_Navigation;
%let sumpgm=/Projects/SESUG2013/Example_5a_Summary;
```

Added this **%let** statement as shortcut to server name and port and URL

```
  %let tsession=http://&_SRVNAME.:&_SRVPORT.&_URL.?;
```

## Example_5a_Summary

Added global Macro definitions

```
   %global rep_stat year country;
   %let tsession=http://&_SRVNAME.:&_SRVPORT.&_URL.?;
```

Changed Title statements for navigation to point to the stored processes.

```
Title4 j=c h=2
      '<A HREF="'
      "%SUPERQ(TSESSION)"
      '&_ACTION=EXECUTE'
      '&_program=' "&sumpgm"
      '&repstat='  "Active"
      '&year='     "2013"
       ' Title="You can place text here for a Hint when Hovering" '
                 '>Link to Stored Process Summary </A>'
      h=2 ' | ' h=2
      '<A HREF="'
      "%SUPERQ(TSESSION)"
      '&_ACTION=EXECUTE'
      '&_program=' "&navpgm"
      '&repstat='  "Active"
      '&year='     "2013"
       ' Title="You can place text here for a Hint when Hovering" '
          '>Link to Navigation Stored Process</A>' h=2 ' | ' h=2
      "<A HREF=""&odspath.Report2_Summary.html"" Title=""Link to Third Report""
>Place Holder</A>"
      h=2 ' | ' h=2
      'Place Holder for Other Reports'
      h=2 ' | ' h=2
      'Place Holder ';

Title5 h=3 c=red j=left "&rep_stat" j=right "&year";
```

Changed definition of link in Proc Report

```
COUNTRY_NAME='<A HREF="' || "%SUPERQ(TSESSION)" ||
            '&_program='|| "&detpgm" ||
            '&repstat=' || "Active" ||
            '&year='    || "2013" ||
            '&country=' || trim(country) ||
            '&_debug=' || "LOG" ||'"'||"
            Title=""Detail Report"" >"||trim(x)||"</A>";
```

## Example_5b_Detail

Add global macro definitions

```
%global country Year rep_stat;
```

Add Macro for session rather than hardcode server name and port into HREF's

```
%let tsession=http://&_SRVNAME.:&_SRVPORT.&_URL.?;
```

Add Macro definitions for metadata program names.

```
%let navpgm=/Projects/SESUG2013/Example_5_Navigation;
%let sumpgm=/Projects/SESUG2013/Example_5a_Summary;
%let detpgm=/Projects/SESUG2013/Example_5b_Detail;
```

Modify Title statements for navigation bar.

```
Title4 j=c h=2
      '<A HREF="'
      "%SUPERQ(TSESSION)"
      '&_ACTION=EXECUTE'
      '&_program=' "&sumpgm"
      '&repstat='  "Active"
      '&year='     "2013"
      '&_debug='   "LOG" '"'
      ' Title="You can place text here for a Hint when Hovering" '
                  '>Link to Stored Process Summary </A>'
      h=2 ' | ' h=2
      '<A HREF="'
      "%SUPERQ(TSESSION)"
      '&_ACTION=EXECUTE'
      '&_program=' "&navpgm"
      '&repstat='  "Active"
      '&year='     "2013"
      '&_debug='  "LOG" '"'
      ' Title="You can place text here for a Hint when Hovering" '
                  '>Link to Navigation Stored Process</A>'
      h=2 ' | ' h=2
      "<A HREF=""&odspath.Report2_Summary.html"" Title =
       ""Link to Third Report"" >Place Holder</A>"
      h=2 ' | ' h=2
      'Place Holder for Other Reports'
      h=2 ' | ' h=2
      'Place Holder ';

Title5 h=1 j=left "&rep_stat" j=right "&year";
```

Create Stored Processes from Modified Code:

1. In each program click on properties, then create stored process
2. Name the Stored Processes
    a. Example_5_Navigation

b. Example_5a_Summary
c. Example_5b_Detail

Open up the Stored Process Web Application and navigate through our newly created example.

Note that if you use this link you will be prompted for userid and password

**http://localhost:8080/SASStoredProcess/do?**

Bit if you use this link you will be taken directly to the application because we set AllowGuest to be true in Management Console

**http://localhost:8080/SASStoredProcess/guest?**

## Example 6 – AJAX, Javascript, and HTML

Our next and final example will show how we can run these stored processes directly from an HTML page or website that you have created. It has provisions for prompting and even a live lookup to the data to assist in prompting and provide valid choices.
We will review and execute this code.

**Ajax** is an acronym for **A**synchronous **J**avaScript **A**nd **X**ML. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Data can be retrieved using the XMLHttpRequest object.

Example_6_HTML utilizes this technique to retrieve the valid country names in a box for the user to choose from. I have created another stored process called “Select_Country” to do this.

```
*ProcessBody;
%global _odsdest _result;
%let _ODSDEST=_webout;
%let _RESULT=STREAM;

options nocenter;
libname _WEBOUT xml ;

proc sql noprint;
create table _WEBOUT.table as select distinct country from
sashelp.prdsale;
quit;
run;

libname _WEBOUT clear;
ods _all_ close;
```

This program will create an XML stream (not a file) with a list of the counties in our PRDSALES dataset on sashelp.
The strange formation of the libname statement is to put the XML out to a stream and not a fixed file. Notice we define the “_WEBOUT” library with the XML engine, then write some randomly named table. Here, the table is named: “table” and the SAS library “_Webout”. The Javascript in Example_6.HTML will execute the stored process and read that XML data as an input stream and populate a text box.

Great site for Javascript and AJAX tutorial

http://www.w3schools.com/ajax/ajax_xmlhttprequest_send.asp

Link to SAS Documentation of _webout.table statement:

http://www.sas.com/offices/europe/uk/support/sas-hints-tips/stored.html

Examine the HTML 'Example6.html' with notepad. The first division has a navigation bar similar to what we built in Example 5. The anchor tag and HREF property to create a link to a stored process looks as we would expect:

```
<a
href="http://localhost:8080/SASStoredProcess/guest?&_program=/Projects/Sesug2013/Example_5
_Navigation&year=2013&repstat=html" Title='Hard Coded Link to Navigation STP'>Navigation
Bar1</a>
```

The second division has some JavaScript to create functions that will utilize the XMLHttpRequest object to retrieve data from the STP –Select_Country. The rest of the HTML creates a small table and several examples of input for us to use. Upon completion of input and pressing the Submit button the HTML page sends its request to the SAS Stored Process Server. Notice the use of "POST" rather than "GET". SAS seems to run faster with POST but works as well.

## APPENDIX

This section describes how to configure the SAS Stored Process Web Application to allow guest use without passing a username and password. This is a link to the SAS 9.3 documents that describe these procedures.

http://support.sas.com/docume`ntation/cdl/en/stpug/62758/HTML/default/viewer.htm#n0ax7wadghvldwn1bbarfuly6adl.htm

Starting with SAS 9.2, users can run stored processes without having to log on. A guest user name can be defined to run stored processes under a fixed account. The guest user name and password are specified in the SAS Stored Process Web Application initialization parameters.

To enable guest access, the SAS Stored Process Web Application initialization parameter AllowGuest must be set to true. Use the Configuration Manager in SAS Management Console to set this parameter.

Expand the Configuration Manager group on the Plug-ins tab in SAS Management Console.
Right-click the Stored Process Web App 9.3 node and select Properties.
In the Properties dialog box, click the Advanced tab.
Double-click the property value for the AllowGuest property, and change the value to true in order to grant guest access to the Anonymous Web User.

After you modify the advanced properties for the SAS Stored Process Web Application in the Configuration Manager, you must stop the Web application server, restart SAS Remote Services, and then start the Web application server.

You may access the application with this type of link as we have done in our application.
If ALLOWGUEST is true, then the following command will bypass the logon screen and take you directly to the application using the 'webanon' username and password

http://yourserver.com:8080/SASStoredProcess/guest?

Here are some screen shots from the Management Console.

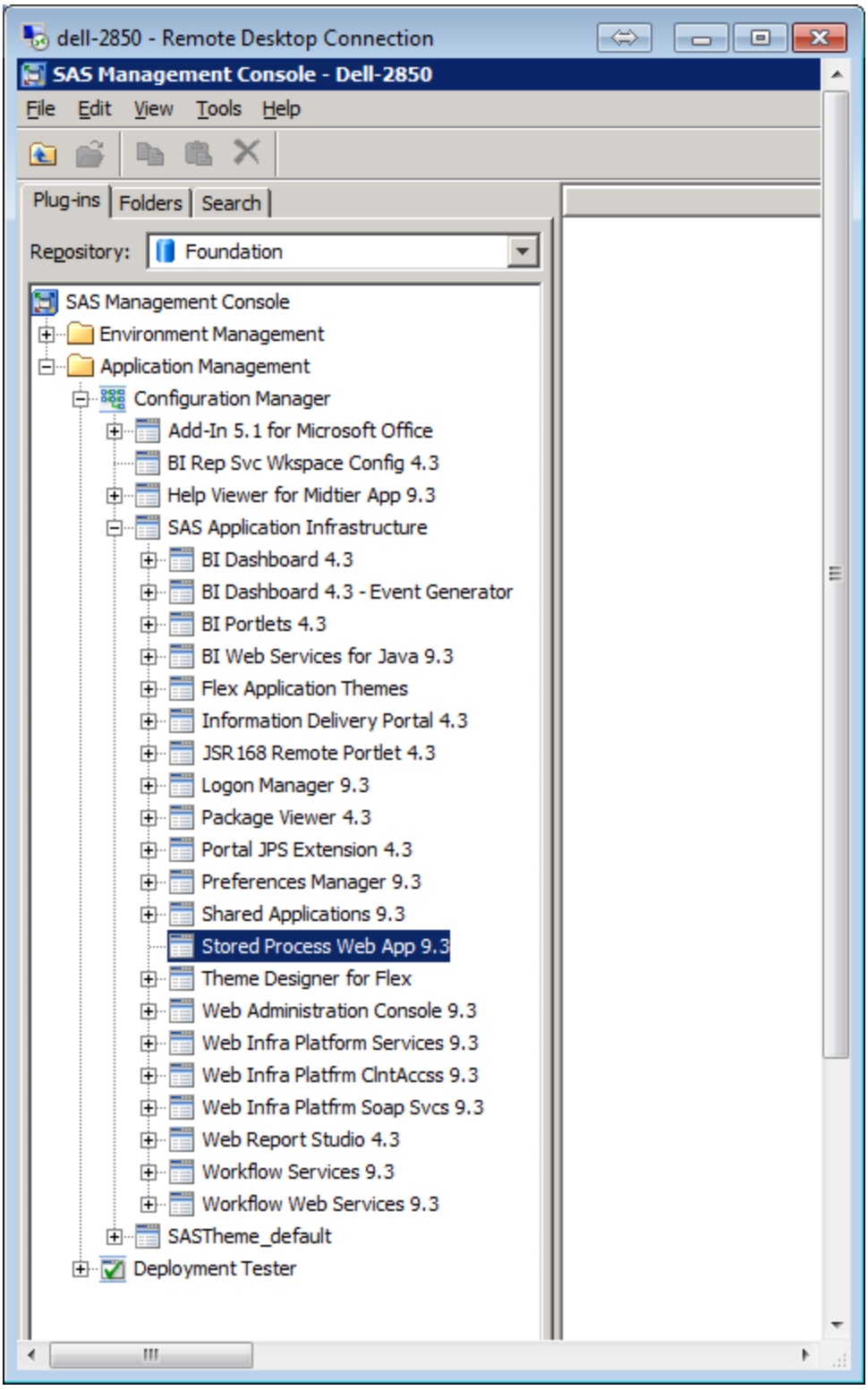



**Figure 9. Stored Process Web Application – Configuration Management**

Right Click and Choose Properties
Then on advanced Tab you will see settings for guest
Add an entry for AllowGuest and set value to be true

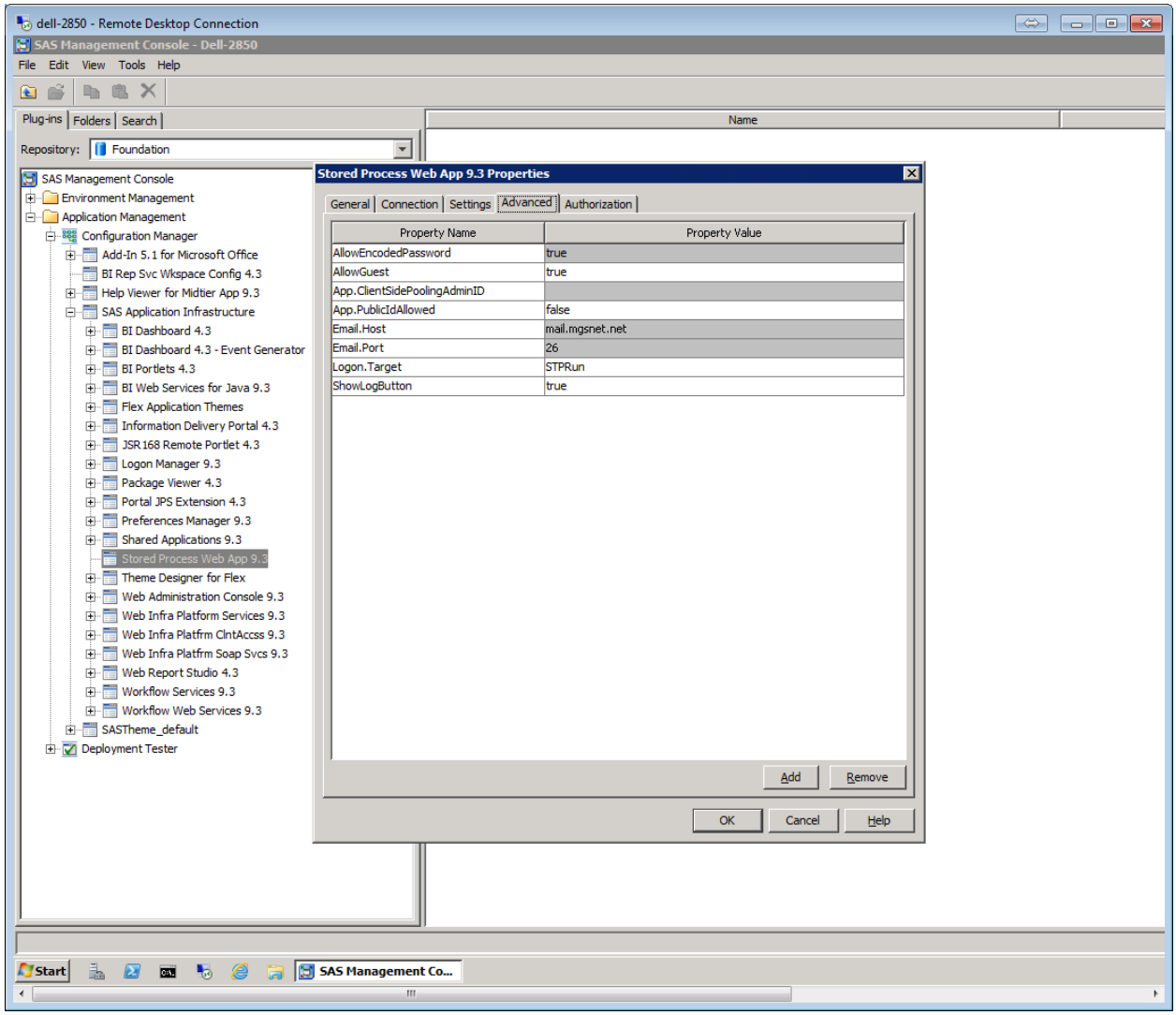



**Figure 10. Properties of the Stored Process Web App 9.3**

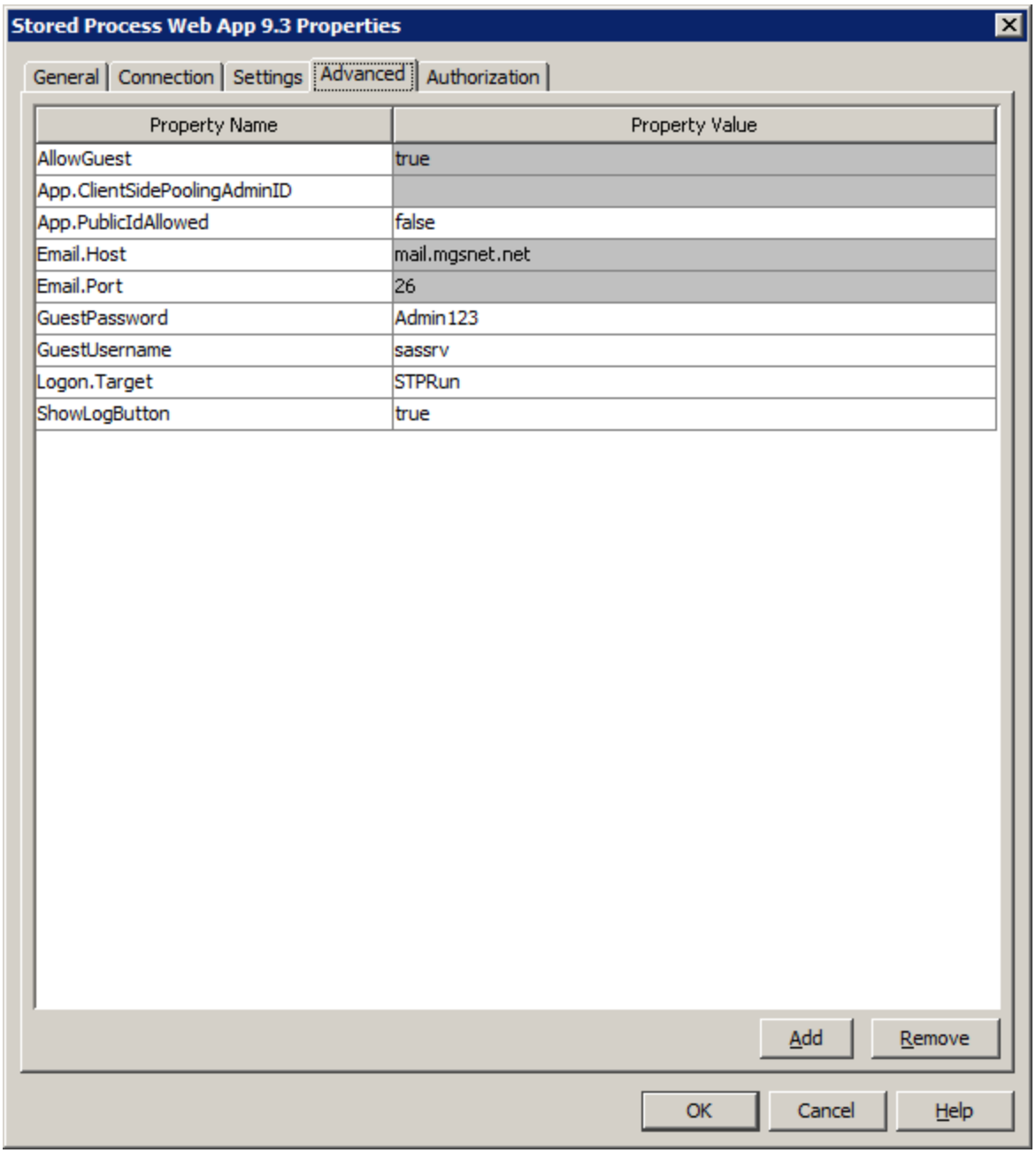



**Figure 11. Setting of AllowGuest property and adding GuestPassword and GuestUsername**

## CONCLUSION

By combining Proc Report's capability of utilizing HTML anchor tags and the SAS Stored Process Web Application's ability to run a stored process we can develop dynamic drill down reports that can be surfaced on a web page. This technique involves the combination of ODS and HTML. Relatively simple HTML can provide the user with a prompting framework. More advanced HTML, JavaScript, and Ajax can provide the user with a dynamic prompting.

## REFERENCES

Carpenter, Art. *Carpenter's Complete Guide to the SAS® Proc REPORT Procedure*, SAS Institute, Cary, NC. 2007

Hamilton, Jack. *Using the Compute Block in Proc Report ,* Proceedings of the 13th Annual Western Users of SAS Software, 2005

Haworth, Lauren E., Zender, Cynthia L., and Burlew, Michele M. *Output Deliver System: The Basics and Beyond.* SAS Institute, Cary, NC. 2009

Mason, Phillip, *Using SAS® Stored Processes with JavaScript and Flash to Build Web-Based Applications*, SAS Global Forum 2011

SAS Institute Inc. *SAS® 9.3 Stored Processes: Developer's Guide*. SAS Institute Inc., Cary, NC. 2011

http://www.w3schools.com/ - Reference for Javascript and XML online

http://www.sas.com/offices/europe/uk/support/sas-hints-tips/stored.html - SAS UK Stored Process Hints and Tips

## CONTACT INFORMATION

Comments, questions, and additions are welcomed.
Contact the authors at:

Michael G. Sadof
MGS Associates, Inc.
Bedford, NH
mgs@mgsnet.net

Louis T. Semidey
The Semidey Group
San Francisco, CA
ltsemidey@aol.com

## TRADEMARKS

## JAVA – AJAX CODE SAMPLE

```
<!DOCTYPE html PUBLIC  "-//W3C//DTD XHTML 1.0 Transitional//EN"  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml" >

<title>Stored Process JavaScript Example</title>

<!-Sample Call to STP will look like this:
        http://localhost:8080/SASStoredProcess/guest?&_program=/Projects/SESUG2013/Sadof/Example_5_Navigatio
n&year=2013&repstat=html"
-->

<style type= "text/css">
        DIV.one {background:#00FF00}
        DIV.two {background:#ADD8E6}
</style>

<head >
  <h1 style="font-size:19pt;color:#0066CC  ">Example 6 Execute Stored Process from HTML</h1>
</head>

<div class= "one">
 <h2 style="font-size:14pt;color:#0066CC  ">Division1: Standard Navigation Bar with Hardcoded Links to Stored
Processes</h2>
    Enter Your Selection: |
   <a
href="http://localhost:8080/SASStoredProcess/guest?&_program=/Projects/SESUG2013/Sadof/Example_5_Navigation
&year=2013&repstat=html" Title='Hard Coded Link to Navigation STP'>Navigation Bar1</a> |
   <a
href="http://localhost:8080/SASStoredProcess/guest?&_program=/Projects/SESUG2013/Sadof/Example_5a_Summary
&year=2013&repstat=html" Title='Hard Coded Link to Summary STP'>Summary Report</a> |
   <a href="http://localhost:8080/SASStoredProcess/guest?" Title='Link to SAS Web APp'>SAS Stored Process Web
App</a> |

<p> </p>
<p> </p>
</div>

<hr size="2" width="100%" align="center" noshade>

<div class= "two" >
<h2 style="font-size:14pt;color:#0020C2  ">Division2: Javascript Enabled Static and Dynamic Choices</h2>
<body>

<!-   These are the Javascript Functions -->

<script type="text/javascript">

 //Get A value into a variable ;
 function getval(sel) {var value = sel.options[sel.selectedIndex].value;}
```

```
 //Load an XML Table into the Variable txt ;

 function loadXMLDoc(){
  var xmlhttp,txt,x,i;
  xmlhttp=new XMLHttpRequest();
  xmlhttp.onreadystatechange=function()  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {xmlDoc=xmlhttp.responseXML;
    txt="";
    x=xmlDoc.getElementsByTagName("COUNTRY");
    var Animals = document.getElementById("Animals");
    for (i=0;i<x.length;i++) {
            txt=txt + x[i].childNodes[0].nodeValue + " ";

        var Entry = document.createElement("option");
        Entry.text = x[i].childNodes[0].nodeValue;
        Animals.add(Entry ,null);

     }
    document.getElementById("myCountry").value=txt.trim();
    };
   }

xmlhttp.open("GET",'http://localhost:8080/SASStoredProcess/guest?&_program=/Projects/SESUG2013/Sadof/SELECT
_COUNTRY', true);
  xmlhttp.send();
}

</SCRIPT>
<!- This form when submitted will execute the following action with all form variables eg: &countrys -->

<form id="frmSearch" name="frmSearch" method="POST" action="http://localhost:8080/SASStoredProcess/guest?"
             >

<table width= "580"  border= "1">
<th width="178" height="34" scope="100%" bgcolor=  "#FFCC66" >Your Variables</th>
<th width="170"             scope="col"  bgcolor=  "#FFCC66" >Your Choices</th>

<tr bgcolor=  "#FF33FF"  ></tr>
 <th colspan=  "13"   align=  "left"    scope= "col" ></th>

 <tr>
 <td>Choose Your Example</td>
 <td>
  <select id="_program"  name="_program"  onchange="getval(this)">
   <option value= "/Projects/SESUG2013/Sadof/Example_4_Navigation" >Example 4 Navigation </option>
   <option value= "/Projects/SESUG2013/Sadof/Example_4a_Summary"   >Example 4a Summary   </option>
   <option value= "/Projects/SESUG2013/Sadof/Example_4b_Detail"    >Example 4b Detail    </option>
   <option value= "/Projects/SESUG2013/Sadof/Example_5_Navigation" >Example 5 Navigation </option>
   <option value= "/Projects/SESUG2013/Sadof/Example_5a_Summary"   >Example 5a Summary   </option>
   <option value= "/Projects/SESUG2013/Sadof/Example_5b_Detail"    >Example 5b Detail    </option>
  </Select>
</td>
</tr>

<tr>
```

```
<td>Reporting Year</td>
<td><input name= "year"  type= "text"  id= "year" ></td>

</tr>

<td>Reporting Period</td>
<td><select name= "rep_stat" >   //This populates &repstat ;
<option value= "None" >None</option>
<option value="ANNUAL" selected >ANNUAL</option>
<option value="QUARTER">QUARTER</option>
<option value="SEMESTER">SEMESTER</option>

//Now lets populate country with AJAX dropdown ;
<tr>
  <td>Choose Country for<br>Detail Report:</td>
  <td>
    Click this button to populate<br>dropdown box of countries using AJAX<br><br>
    <button type="button" onclick="loadXMLDoc()" >**</button>
    <select name="Country" id="Animals" style="white-space:pre-wrap; font-weight:bold; width:100px; background-
color:#FCAAD8; color:#000000"  >
    </select>
  </td>
</tr>

<tr>
 <td>
   This Box Displays List of Countries<br>Retrieved using AJAX:<br>
 </td>

<td>

   <textarea  id="myCountry"  value='clickme'
              style="white-space:pre-wrap; width:100px; height:50px; background-color:#FCF5D8; color:#0000FF;">
   </textarea>
</td>
</tr>

</Form>
</table>
<p>      </p>
<p>      </p>

 <input TYPE="submit" VALUE="Submit Program" Title="Submits Program with Form Parameters"  formtarget="DOG"
>

</div>
</body>

</html>
```