Paper 011-2012

# Mobilizing Your SAS® Business Analytic Reports

Falko Schulz, SAS Institute Inc., Brisbane, Australia

## ABSTRACT

This paper explains the steps involved in getting your SAS® Stored Processes published to mobile devices such as iPhones and iPads. Specific HTML markup and JavaScript techniques allow a seamless integration of SAS® output to be rendered on mobile devices. Considering the smaller screen real estate and input gestures, an intuitive touch-and-swipe user interface is used. This mobile interface allows the user to quickly browse for the desired stored process and view the rendered output. The paper provides an example toolkit including source code that can be used to get your business reports mobilized.

## INTRODUCTION

HTML5 provides mobile device users richer Web applications and improved usability. With HTML5, advanced Web application features are available in all mobile browsers supporting the markup language, using the same standard syntax and displaying the same standard behavior. With the growing demand for mobility, enterprises require mobile support from any BI vendor. In comparison to 5 years ago, there is a real demand to provide mobile business intelligence to businesses.

Organizations have spent a significant amount of time to develop and grow their existing SAS business intelligence environment. This paper will explain the steps involved in getting existing SAS Stored Processes published to mobile devices. There is no need to re-develop existing SAS reports although the author will explain how to enhance reports to fully utilize the HTML5 standards (for example, using location-based services) and take advantage of some specific mobile device features (for example, touch-and-swipe).

## APPLICATION DESIGN

In order to provide a touch-optimized mobile interface, a Web application named “Mobile Dashboard” has been written by the author. This Web application uses advanced JavaScript frameworks jQuery and jQTouch. Both help greatly in developing user interfaces for popular mobile devices. The written JavaScript code stays lightweight, themeable, and flexible. Instead of writing unique apps for each mobile device or OS, the jQTouch mobile framework will allow you to design a single highly branded and customized Web application that will work on all popular smartphone and tablet platforms.

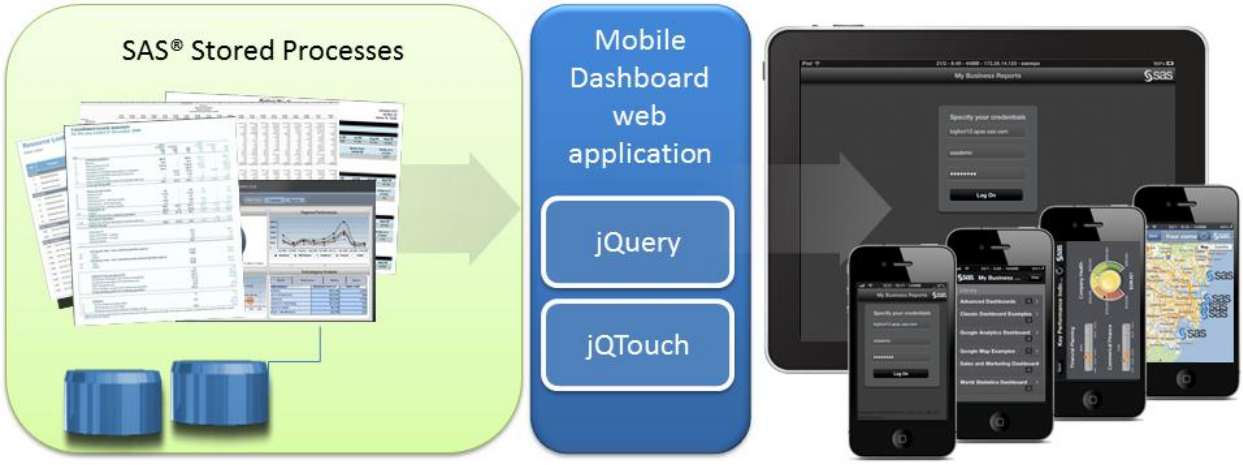
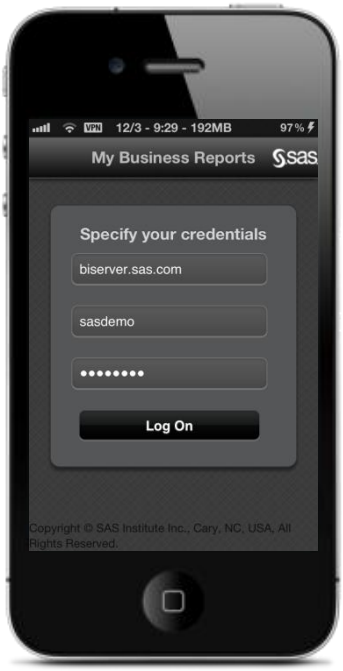



**Figure 1: Application Diagram**

The application also uses the SAS Stored Process Web application, which is part of SAS® BI Server. The SAS Stored Process Web application provides a Web-based access to SAS Stored Processes. Reports render fine on most

desktop browsers (for example, Microsoft Internet Explorer or Firefox), but it's difficult to use on tablets or smartphones due to the small screen design as well as touch screen.




## USER INTERFACE

The application provides a mobile enabled logon screen. The logon credentials are used to authenticate against the SAS Stored Process Web application.

Once logged in, the user can browse registered SAS Stored Processes by tapping a dashboard folder. The interface slides folder or report views as you would expect from native applications. A selected report will be rendered using the SAS Stored Process Web application.


In addition to typical animated page transitions, back buttons, etc., the application also supports orientation changes and will re-render reports using correct screen sizes if the device is rotated.

## DEPLOYMENT

The Mobile Dashboard application provides an enterprise archive (.ear file) and can therefore easily be deployed in all Web application servers supported by SAS BI Server (for example, JBoss). You can download the current version of the Mobile Dashboard Web application from http://www.sascommunity.org/wiki/Mobilizing_Your_SAS_Business_Analytic_Reports. If you are using the popular JBoss application server, the deployment would be as simple as copying the `sgf.mobile.toolkit.ear` file to the `deploy_sas` folder of the server (for example, SASServer1) hosting the SAS Stored Process Web application (`sas.storedprocess9.3.ear`).

Make sure you explode (=extract) the archive, as it will make things easier when modifying files during the development. Once deployed and exploded, you should have a folder structure such as the one shown in Figure 2: Web Application Folder Structure.

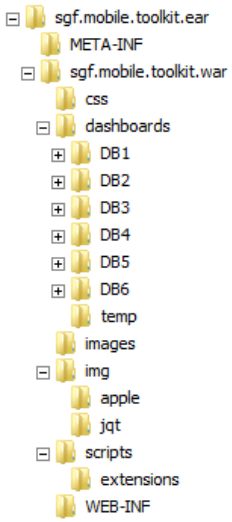



**Figure 2: Web Application Folder Structure**

The application utilizes SAS Stored Processes to render the user interface. This stored process needs to be registered in the SAS® Management Console following the steps outlined below:

1. Open SAS Management Console and open the SAS Folders tab. Navigate to the folder you wish to use and right-click on it to select "New Stored Process."
2. Enter a name for the stored process (for example, "dashboard.toolkit").
3. Select the SAS® Stored Process Server as the SAS server.
4. Select or Register the source repository for the `stp_dashboard.sas` file. This file resides in the `sgf.mobile.toolkit.war` directory (subdirectory in `sgf.mobile.toolkit.ear`).
   **Note**: The author decided to store the `stp_dashboard.sas` inside the Web archive. Although it's not recommended in a production environment, it makes development and sharing easy. In a production

environment, you might consider moving this file to another location where you have all your other SAS Stored Processes stored.

5. Select `Streaming` as output type.
6. Click on Finish to save this new SAS Stored Process registration.
7. As a final step, open the `stp_dashboard.sas` using your preferred SAS editor and update the HOME_DIR macro variable in line 9 with the full physical path of the home directory of this file. Unless you moved this file, this should point to the `sgf.mobile.toolkit.war` directory.

The application also comes with a properties file (`config.xml`) located in the same directory as the `stp_dashboard.sas` file. You will need to open this file using a text editor and update options as required. Be sure to maintain the valid XML structure while editing. The following list explains each configuration value and its purpose:

1. `APPLICATION_TITLE` = the main banner text shown in the logon screen.
2. `APPLICATION_SUBTITLE` = the subtitle text shown in the logon screen.
3. `_ODSSTYLE` = the default SAS style used to render SAS Stored Processes.
4. `APPLICATION_WEB_SERVER` = the full qualified name (or IP address) of your application server.
5. `APPLICATION_WEB_PORT` = the port number of your application server (for example, `8080`).
6. `APPLICATION_DEFAULT_USERNAME` = the default user name for the logon screen.
7. `APPLICATION_DEFAULT_PASSWORD` = the default password for the logon screen (optional).
8. `APPLICATION_STP_PATH` = the full path pointing to the SAS Stored Processes registered above.
9. `APPLICATION_WEB_ROOT` = the context name of the Web application. Unless you changed the Web application name in `sgf.mobile.toolkit.ear\META-INF\application.xml` you don't have to change this value.
10. `APPLICATION_AUTO_LOGON` = switch for performing an auto logon (this feature has not been implemented yet).

To finalize the deployment, restart your application server and try accessing the main application URL using http://machine-name:8080/SGFMobileDashboard . The application logon screen should be presented, allowing you to adjust user name and password if required.

## BEHIND THE SCENES

The core of the application is supported by the stored process `stp_dashboard.sas` registered in the previous step. It uses SAS code to generate the application user interface as well as acting as an action controller in the sense of a model viewer controller (MVC) application. Based on user input (for example, a user taps on a report folder) the application calls itself passing information about the recent action. Depending on the action, different parts of the user interface are rendered. The diagram shown in Figure 3: Application Flow Diagram helps understanding of the application architecture.


**Figure 3: Application Flow Diagram**

Here is an explanation of the steps:

1. The initial list of report folders is read from the database and rendered as list component.
2. The user selects a folder, and containing reports are listed.
3. The user selects a report, and the report content is rendered using the SAS Stored Process Web application.

The structure of the underlying `stp_dashboard.sas` reflects the above application flow. The following code snippets explain the main sections of the SAS code.

```
/* read dashboard configuration files */
libname CONF xml "&HOME_DIR\dashboards\config.xml";
data WORK.DASHBOARD;
     set CONF.DASHBOARD;
run;
data WORK.INDICATOR;
     set CONF.INDICATOR;
run;
proc sort data=WORK.INDICATOR out=WORK.INDICATOR;
    by title;
run;
proc sql;
     create table dashboard as
     select * from CONF.DASHBOARD a
     left join (
         select DASHBOARD as did, count(DASHBOARD) as INDICATORS
         from indicator
         group by dashboard
     ) b
     on a.id = b.did
     order by title
;quit;

/* read application configuration files */
libname CONF xml "&HOME_DIR\config.xml" xmltype=msaccess;
data WORK.CONFIG;
     set CONF.CONFIG;
     CALL SYMPUT(param,trim(value));
```

The application initializes with reading application settings from configuration files. There is one main configuration file located in the main home directory as well as one dashboard configuration file containing details about registered reports.

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
- <dataroot>
  - <CONFIG>
      <VALUE>My Business Reports</VALUE>
      <PARAM>APPLICATION_TITLE</PARAM>
      <LABEL>application title</LABEL>
      <TYPE>text</TYPE>
      <VALUES/>
    </CONFIG>
  - <CONFIG>
      <VALUE>SAS Institute, 2012</VALUE>
      <PARAM>APPLICATION_SUBTITLE</PARAM>
      <LABEL>application subtitle</LABEL>
      <TYPE>text</TYPE>
      <VALUES/>
    </CONFIG>
  - <CONFIG>
      <VALUE>minimal</VALUE>
      <PARAM>_ODSSTYLE</PARAM>
      <LABEL>default ODS style</LABEL>
      <TYPE>combo</TYPE>
      <VALUES>AMODefault,Analysis,Astronomy,Bank
    </CONFIG>
  - <CONFIG>
      <VALUE>172.26.14.148</VALUE>
      <PARAM>APPLICATION_WEB_SERVER</PARAM>
      <LABEL>server host name</LABEL>
      <TYPE>text</TYPE>
      <VALUES/>
```

```
%else %do;

        data _null_;
            file _webout;
            put '<!doctype html>';
            put '<html>';
            put "<head><title>&APPLICATION_TITLE.</title>";
            put "<meta charset='utf-8'>";


            put "<style type='text/css' media='screen'>@import '&
            put "<style type='text/css' media='screen'>@import '

            put "<script src='&APPLICATION_WEB_ROOT./scripts/jqto
            put "<script src='&APPLICATION_WEB_ROOT./scripts/jqu
            put "<script src='&APPLICATION_WEB_ROOT./scripts/jqto
            put "<script src='&APPLICATION_WEB_ROOT./scripts/exte
            put "<script src='&APPLICATION_WEB_ROOT./scripts/mob
            put "<script type='text/javascript' charset='utf-8'>
            put "    var jQT = new $.jQTouch({";
            put "        icon: '&APPLICATION_WEB_ROOT./images/ap
            put "        icon4: '&APPLICATION_WEB_ROOT./images/a
            put "        addGlossToIcon: true,";
            put "        startupScreen: 'images/iPhone_320x480.pn
            put "        statusBar: 'black-translucent',";
            put "        themeSelectionSelector: '#jqt #themes ul
            put "        preloadImages: []";
            put "    });";
            put "    window.onorientationchange = function(){";
```

The application executes the main macro ***%renderDoc;*** which conditionally executes different parts of the code depending on the action taken by the user. If there is no action passed (for example, during the first launch of the application), the home screen is rendered displaying a list of report folders.

As some parts of the user interface are static, simple ***PUT*** statements are used to generate the HTML5 Web page.

There are also references to the JavaScript frameworks jQTouch and jQuery in order to support mobile device features (for example, home icons, status bar, splash screens, etc.).

```
%else %if %superq(indicatorid) ne %then %do;

        data _null_;
            file _webout;
            set WORK.INDICATOR;
            where dashboard eq "&dashboardid." and id eq "&indicator
            baseURL = "&_THISSESSION" || '&_PROGRAM=' || "&_PROGRAM
            widgetid = "W_" || trim(dashboard) || "_" || trim(id);
            params = tranwrd(parameter,"|","&");

            put '<div id="' dashboard +(-1) '_' id +(-1) '" class="e
            put '<div class="toolbar">';
            put "<a href='http://www.sas.com' target='_new'><img sr
            put '<h1>' title +(-1) '</h1>';
            put '<a class="button back" href="#">Back</a>';
            put '<a class="button" href="#" onclick="' widgetid +(-
            put "<img src='&APPLICATION_WEB_ROOT./images/refresh.png
            put '</div>';
            put '<ul class="edgetoedge">';
            put '<iframe name="' widgetid +(-1) '" ID="' widgetid +
            put ' onload="if (' widgetid +(-1) '.location.href ==
            put 'var indiUrl=' "'" baseURL +(-1) '&indicatorid=' i
            put '&_width=' "' + viewport().width + '" '&_height='
            put widgetid +(-1) ".location.href=indiUrl;console.info
            put "showLoadingInfo(true);" @;
            put '}" scrolling="no" marginwidth="0" marginheight="0"
            put '</ul>';
            put '</div>';
        run;
```

Once a report is selected, the application renders an IFRAME tag in order to embed the report content. This is achieved by referring to the SAS Stored Process Web application.

The application also monitors events such as `onLoad` or `onOrientationChange`. This way, specific JavaScript code can be executed to handle such an event. If for example the user rotates the device, the application needs to re-render the report content taking into account the new landscape/portrait screen width and height.

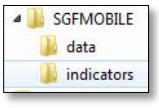
## FIRST MOBILE REPORT

The Mobile Dashboard application comes with a set of standard dashboards or reports. This section will explain the steps involved in getting your own first Hello World mobile report registered and delivered to your mobile device.

Reports are organized in folders underneath the `sgf.mobile.toolkit.war/dashboards` folder as shown in Figure 4: Dashboard Folder Structure. Each folder contains two subfolders:

1. `data` = contains temporary or staging data. This directory is often used to store pre-summarized data in order to improve performance of the reports. You can reference this folder by using the library `DATA`. The use of this library is optional.
2. `indicators` = contains one .sas file for each indicator or report. The name of the .sas file is registered in the main dashboard configuration file.

In order to create our own mobile report, you need to create the physical folders first. For demonstration purposes, we created the following folder structure underneath the dashboards folders:




You will also need to create one file in `SGFMOBILE/indicators` named `helloworld.sas`. This file will contain the actual SAS DATA step used to generate the report.

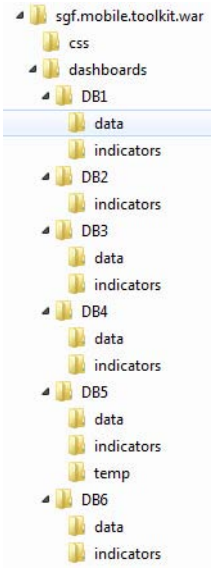



**Figure 4: Dashboard Folder Structure**

The content of this first .sas file will be very simple so we can focus on the main report registration. For demonstration purposes, we will simply list the content of the standard `SASHELP` data set `CLASS`. If you are not familiar with writing your own SAS code, you can also use tools such as SAS® Enterprise Guide® and its great code wizards.

```
ODS HTML body=_webout (no_top_matter no_bottom_matter) path=&_tmpcat (url=&_replay)
parameters=("drilldown"="any" "drilltarget"="_self")
attributes=("ID"="GIDX") style=styles.&_ODSSTYLE;

GOPTIONS xpixels=&_width ypixels=&_height NOTRANSPARENCY device=UPNGT;

goptions nodisplay;

proc print data=sashelp.class;
run;

ODS HTML CLOSE;
ODS LISTING;
```

**Figure 5: helloworld.sas**

Note that `_width`/`_height` macro variables are provided by the program automatically. Those variables will also change if the device is rotated. If you are generating graphical output, it's advisable to use TRANSPARENCY and UPNGT device output to maintain the application main background.

In order to finally get our new report listed in the application it needs to be registered in the `sgf.mobile.toolkit.war/dashboards/config.xml`. The structure of the XML document is shown below:

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
- <TABLE>
      <!-- World Statistics Dashboard -->
  - <DASHBOARD>
      <id>DB1</id>
      <title>World Statistics Dashboard</title>
      <description Missing=" "/>
    </DASHBOARD>
  - <INDICATOR>
      <id>DB1IND1</id>
      <dashboard>DB1</dashboard>
      <title>Broadband Penetration Percentage</title>
      <description Missing=" "/>
      <source>internet_stats.sas</source>
      <parameter Missing=" "/>
      <style Missing=" "/>
    </INDICATOR>
  - <INDICATOR>
      <id>DB1IND2</id>
      <dashboard>DB1</dashboard>
      <title>Largest cities in North Carolina, US</title>
      <description Missing=" "/>
      <source>nc_cities.sas</source>
      <parameter Missing=" "/>
      <style>education</style>
    </INDICATOR>
```

**Figure 6: dashboards/config.xml**

There are two types of XML tags:

1. `<DASHBOARD>` = describes a dashboard or folder. This tag requires nested tags `<ID>`, `<TITLE>` and `<DESCRIPTION>`. Make sure the ID value is unique across this document.
2. `<INDICATOR>` = describes a report or indicator. This tag requires nested tags `<ID>`, `<TITLE>`, `<DASHBOARD>`, `<DESCRIPTION>`, `<SOURCE>`, `<PARAMETER>` and `<STYLE>`. In order to register an indicator for a specific folder, the dashboard ID is specified for the `<DASHBOARD>` value. The `<SOURCE>` value refers to the .sas file located in the `indicators` folder.

In order to register our `helloword.sas` report in the `SGFMOBILE` folder, we would need to add the following XML tags to the `config.xml` document:

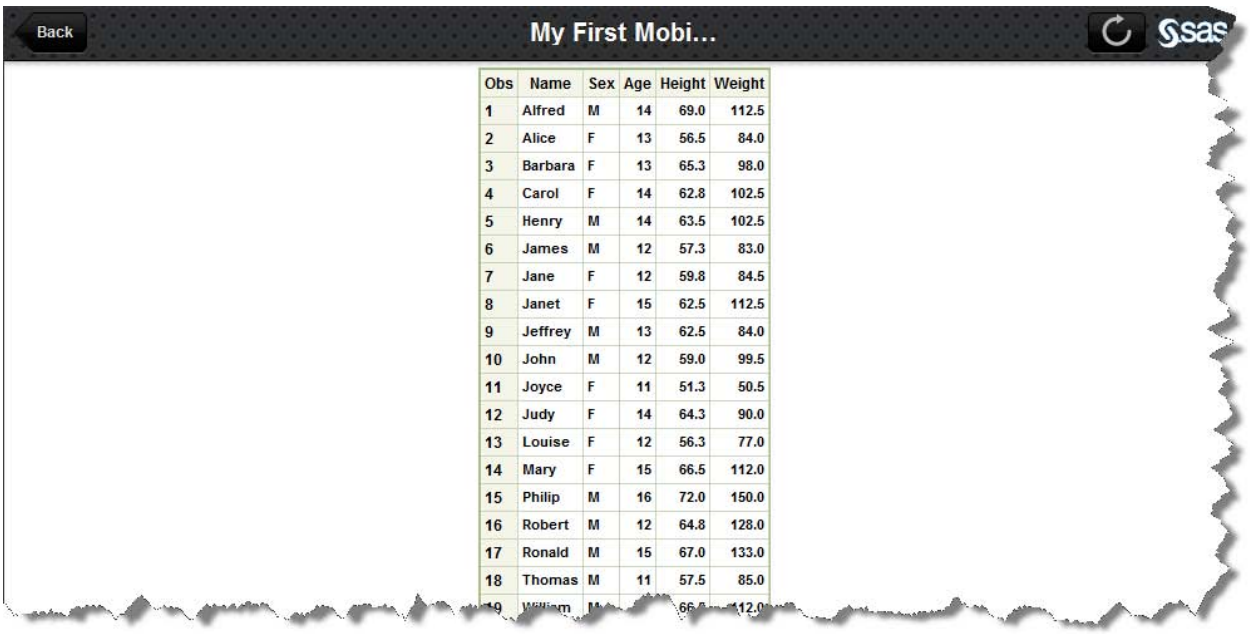
```
- <DASHBOARD>
      <id>SGFMOBILE</id>
      <title>SGF Mobile Reports</title>
      <description Missing=" "/>
  </DASHBOARD>
- <INDICATOR>
      <id>SGFIND1</id>
      <dashboard>SGFMOBILE</dashboard>
      <title>My First Mobile Report</title>
      <description Missing=" "/>
      <source>helloworld.sas</source>
      <parameter Missing=" "/>
      <style>Meadow</style>
  </INDICATOR>
```

**Figure 7: Registration helloworld.sas**

To verify that the registration was successful, simply refresh the browser or restart the application. A new folder “SGF Mobile Reports” should appear in the list:


**Figure 8: Updated List of Reports**

After tapping on the folder and on the report itself, the report should be rendered accordingly:




| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Alice | F | 13 | 56.5 | 84.0 |
| 3 | Barbara | F | 13 | 65.3 | 98.0 |
| 4 | Carol | F | 14 | 62.8 | 102.5 |
| 5 | Henry | M | 14 | 63.5 | 102.5 |
| 6 | James | M | 12 | 57.3 | 83.0 |
| 7 | Jane | F | 12 | 59.8 | 84.5 |
| 8 | Janet | F | 15 | 62.5 | 112.5 |
| 9 | Jeffrey | M | 13 | 62.5 | 84.0 |
| 10 | John | M | 12 | 59.0 | 99.5 |
| 11 | Joyce | F | 11 | 51.3 | 50.5 |
| 12 | Judy | F | 14 | 64.3 | 90.0 |
| 13 | Louise | F | 12 | 56.3 | 77.0 |
| 14 | Mary | F | 15 | 66.5 | 112.0 |
| 15 | Philip | M | 16 | 72.0 | 150.0 |
| 16 | Robert | M | 12 | 64.8 | 128.0 |
| 17 | Ronald | M | 15 | 67.0 | 133.0 |
| 18 | Thomas | M | 11 | 57.5 | 85.0 |
| 19 | William | M | | 66.5 | 112.0 |

**Figure 9: Basic Report Example**

## ADVANCED EXAMPLES

The application also comes with additional examples demonstrating more advanced techniques (for example, cascading prompts or location-based mapping). There are also advanced graph examples (for example, dashboards) originally developed by Robert Allison (see Resources section for other graph examples).

## EXAMPLE CASCADING PROMPTS

You often want to be able to parameterize reports or simply prompt the user to select filter values. A typical requirement is to support cascading prompts where multiple prompts are linked together, meaning a first prompt selection will limit the prompts values available in cascading prompts. The following example shows the SASHELP.PRDSALE sample table with additional filter options on the top of the report:


**Figure 10: Prompted Report**

In order to render the prompts, the underlying indicator needs to generate corresponding select boxes for each category. In this specific example, three select boxes are rendered providing prompt values for Country, Region and Division using a macro `%renderPrompt`.

```
data _null_;
    file _webout;
    put '<div style="text-align:center"><form method="get" action="' "&_THISSESSION" '">';
    put '<input type="hidden" name="_debug" value="0">';
    put '<input type="hidden" name="_program" value="' "&_program" '">';
    put '<input type="hidden" name="frame" value="true">';
    put '<input type="hidden" name="indicatorid" value="' "&indicatorid" '">';
    put '<input type="hidden" name="dashboardid" value="' "&dashboardid" '">';
    put '<input type="hidden" name="_width" value="' "&_width" '">';
    put '<input type="hidden" name="_height" value="' "&_height" '">';
run;

/* generate cascading prompts */
%renderPrompt(name=country, desc=country, label=Country, table=sashelp.prdsale, default=CANADA);
%renderPrompt(name=region, desc=region, label=Region, table=sashelp.prdsale, default=EAST, filte
%renderPrompt(name=division, desc=division, label=Division, table=sashelp.prdsale, default=EDUC

data _null_;
    file _webout;
    put '</form></div>';
run;
```

**Figure 11: Example Rendering Prompts**

The macro `%renderPrompt` simply creates a distinct list of all values and writes out the HTML select boxes:

```
%macro renderPrompt(name=, desc=, label=, table=, default=, filter=);
    %if not %symexist(&name) %then %do;
        %global &name;
        %let &name=&default;
    %end;

    proc sql noprint;
        create table option as
        select distinct &name. as value, &desc. as label
        from &table.
        %if %length(&filter) gt 1 %then %do;
            where &filter
        %end;
        ;
    ;quit;

    data _null_;
        file _webout;
        set option end=last;

        if _n_ eq 1 then do;
            put "&label. <select name='&name.' onChange='parent.showLoadingInfo(true);form.submit();'>";
            call symput("default", trim(value) );
        end;

        if value = "&&&name." then do;
            put '<option value="' value +(-1) '" selected>' label +(-1) '</option>';
            call symput("default", trim(value) );
        end; else do;
            put '<option value="' value +(-1) '">' label +(-1) '</option>';
```

**Figure 12: %renderPrompt Macro**

Note that the `onChange` event is used on the select box to submit the HTML form after a new value is selected by the user. As the form target is the indicator itself, the new value is used as a default filter for cascading prompts.

## EXAMPLE – GEO MAPPING

You can also take advantage of the smartphone device geo location services by providing interactive maps. A popular GIS provider is Google. The example below shows a store at its location and provides a dynamic KPI image in the marker window generated by SAS Stored Processes.

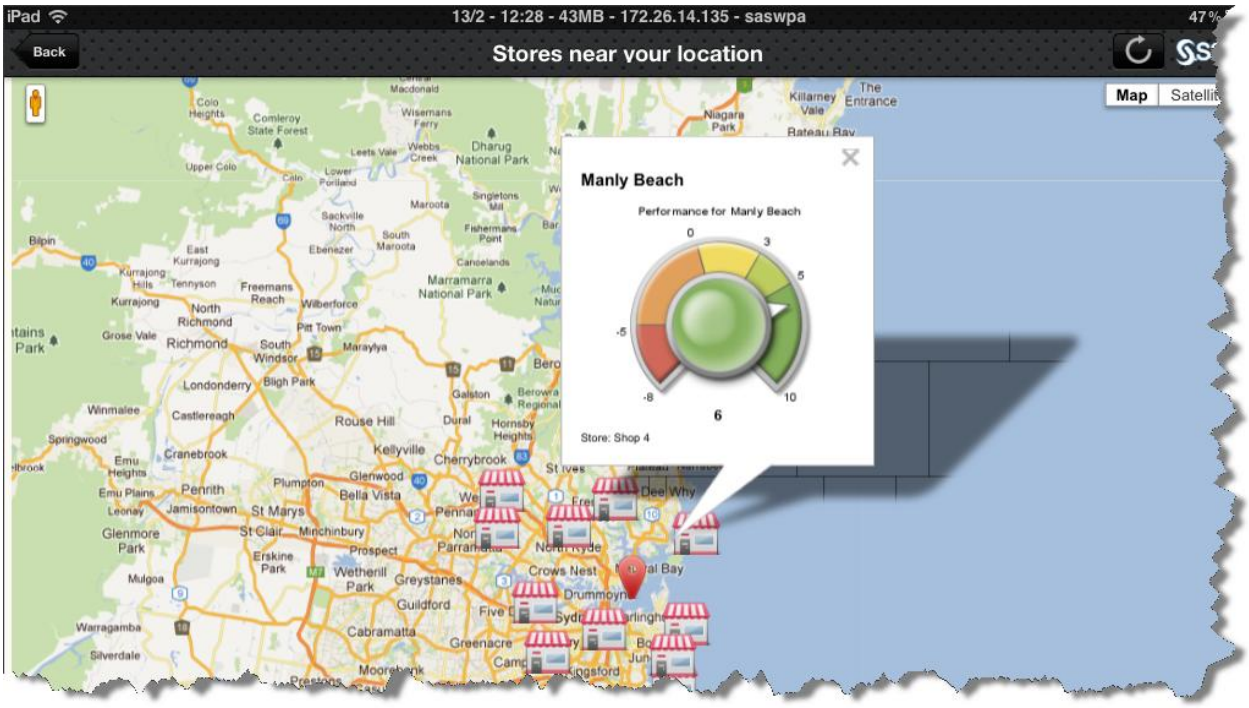



**Figure 13: Map Report Example**

As most parts of this report are static HTML/JavaScript, an external file is referenced and imported using a DATA step. Placeholder variables such as `%WIDTH%` or `%HEIGHT%` are replaced with macro variable values within the same step.

```
data _null_;
     file _webout;
     infile "&HOME_DIR\dashboards\DB4\indicators\google.map.1.html" _infile_=inline;
     input;
     inline=tranwrd(inline, "%ROOT%", compress("http://&APPLICATION_WEB_SERVER.:&APPLI
     inline=tranwrd(inline, "%WIDTH%", compress("&_width.") );
     inline=tranwrd(inline, "%HEIGHT%", compress("&_height.") );
     put inline;
run;
```

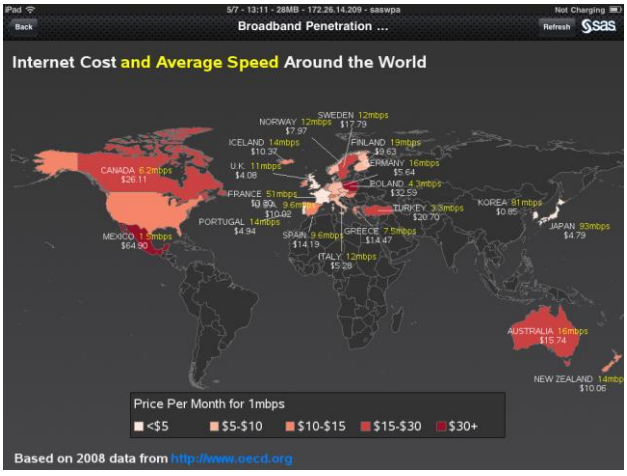
**Figure 14: Import Static HTML File**

For demo purposes, static locations have been used in the following code extract. You can replace this static location array (`locationArr` variable) and reference a database using a DATA step.

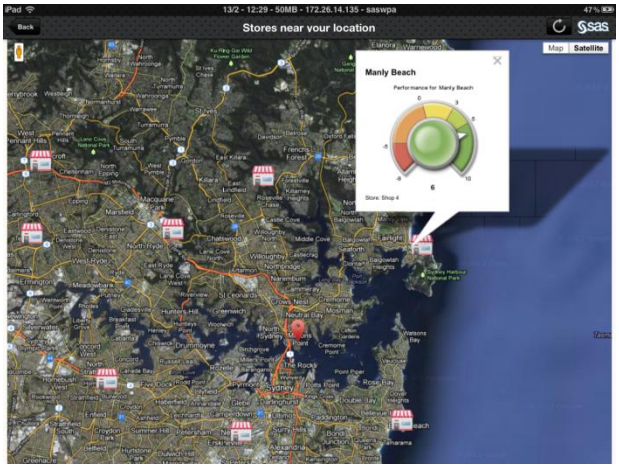
```
<style type="text/css">
    #map_canvas {
        margin: 0;
        padding: 0;
        height: 90%;
    }
</style>
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
<script type="text/javascript">

   var locationArr = [
     ['Bondi Beach', -33.890542, 151.274856, 4, 'shop', 'Shop 1', 12],
     ['Coogee Beach', -33.923036, 151.259052, 5, 'shop', 'Shop 2', 23],
     ['Cronulla Beach', -34.028249, 151.157507, 3, 'shop', 'Shop 3', 43],
     ['Manly Beach', -33.80010128657071, 151.28747820854187, 2, 'shop', 'Shop 4', 13],
     ['Maroubra Beach', -33.950198, 151.259302, 1, 'shop', 'Shop 5', 3],
     ['West Pennant Hills', -33.756315, 151.049652 , 2, 'shop', 'Shop 6', 3],
     ['Parramatta', -33.793985, 151.044159, 2, 'shop', 'Shop 7', 6],
     ['Five Dock', -33.869275, 151.090851, 2, 'shop', 'Shop 8', 3],
     ['Drummoyne', -33.844186, 151.210327, 2, 'gps', 'Home', 3],
     ['French Forrest', -33.764307, 151.188354, 2, 'shop', 'Shop 10', 6],
     ['Campsie', -33.919432, 151.105957, 2, 'shop', 'Shop 11', 3],
     ['Surry Hills', -33.896637, 151.174622, 2, 'shop', 'Shop 12', 5],
     ['North Ryde', -33.791702, 151.132050, 2, 'shop', 'Shop 13', 12]
   ];

 var infowindow = new google.maps.InfoWindow(    {
   content: 'n/a',
   size: new google.maps.Size(100,30),
 });
 var kpiStpUrl = '/SASStoredProcess/do?_program=/Projects/Stored%20Processes/GoogleMapsKPI&request=KPI&_markername=
   function setMarkers(map, locations) {
     var shape = {
         coord: [1, 1, 1, 20, 18, 20, 18 , 1],
         type: 'poly'
     };

     for (var i = 0; i < locations.length; i++) {
       var beach = locations[i];
       var myLatLng = new google.maps.LatLng(beach[1], beach[2]);
       var marker = new google.maps.Marker({
           position: myLatLng,
           map: map,
           icon: new google.maps.MarkerImage('%ROOT%/dashboards/DB4/indicators/' + beach[4] + '.png',
                 new google.maps.Size(100, 41),
                 new google.maps.Point(0,0),
```

**Figure 15: google.map.1.html**

## FURTHER EXAMPLES








## GLOSSARY

- jQuery Mobile – A fast, concise, library that simplifies how to traverse HTML documents, handle events, perform animations, and add AJAX ( http://jquery.com ).
- jQTouch - Zepto/jQuery plug-in for mobile Web development on the iPhone, Android, iPod Touch, and other forward-thinking devices (http://jqtouch.com ).
- JBoss – JBoss Application Server (or JBoss AS) is an open-source Java EE-based application server (http://www.jboss.org ).

## CONCLUSION

SAS provides a very robust platform to deliver mobile reports to smartphones today. Customers have invested heavily in building their business reports, and SAS Stored Processes are a great and simple way of pushing reports out to mobile. Using HTML5, mobile developers can use the same set of technologies they know to build rich Web applications that work across different device types. Combine both and you get your very own rich mobile application.

## RESOURCES


- Robert Allison's SAS/GRAPH® Examples. Available at http://robslink.com/SAS/Home.htm
- Google Maps JavaScript API. Available at http://code.google.com/apis/maps/documentation/javascript


## ACKNOWLEDGMENTS

The author would like to thank Jeremy Rankcom for his help in reviewing this paper and also Robert Allison for kindly providing some great SAS/GRAPH examples.

## CONTACT INFORMATION

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

Falko Schulz
SAS Institute Australia
1 Eagle St
Brisbane, QLD, 4001
Work Phone: 07-3233 320
E-mail: Falko.Schulz@sas.com