

Drill down into your SAS data on the web while using XML, javascript and HTML
Bruce Kayton, San Diego, CA

ABSTRACT

XML is largely touted as the way of the future for sharing data between applications and across platforms. Problem is, that on it's own, XML doesn't present itself all too well. This is particularly true when large quantities of data are presented. This paper discusses how to generate the XML output using SAS and then explore ways to present the data in a browser using HTML, Javascript and Cascading Style Sheets.

INTRODUCTION

This paper discusses a methodology to simulate dynamic drill down into data via a web interface. The technique deploys a standard HTML template with embedded data formatted in XML. The drilldown capability uses javascript to populate selection lists which in turn are used as navigation paths for drilling into the data.

DESIGN PRESENTATION LAYOUT

Layout of the data to be presented will be in a tabular format.

The web page has 3 components:

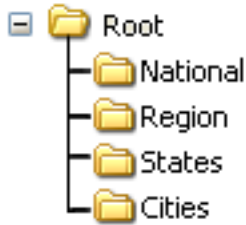
1. Basic HTML code
2. Reference to the XML to be integrated into the document.
3. Javascript to enable data population and drill down navigation.

DETERMINE DRILL DOWN CATEGORIES

Data must be summarized to the appropriate level to accommodate navigation to that level. File subdirectories must be created to accommodate each category of data.

For demonstration purposes we'll create data for 4 levels. These are hypothetical numbers for cars on the road. The levels are National, Regional, State and Cities. Each level has a further 5 categories of car types. All, Compact, Midsize, Standard and Luxury.

Each XML file will contain data for all categories. The subdirectory folders will be constructed as follows:



The National folder will contain a single file national.xml with data summarized at the national level.

The Region folder will contain a single file region.xml with data summarized for the following regions:

- New England
- Middle Atlantic
- East North Central
- West North Central
- South Atlantic
- East South Central
- West South Central
- Mountain
- Pacific

The States folder will contain 9 files named for each of the above regions and will reflect data summarized at a state level for each state within a region

The Cities folder will contain the number of files relative to the states that are represented in each region. i.e. the Pacific region will have:

- Alaska
- California
- Hawaii
- Oregon
- Washington

Each state file then has summarized data for each city occurring in that state i.e. california.xml will contain data for San Francisco, san Jose, Los Angeles, San Diego, Sacramento etc

SUMMARIZE DATA BASED UPON DRILL DOWN CATEGORIES

Use SAS® to summarize data by each category and export output to XML files. You will most likely want to use the FTP engine to dynamically write to your destination folders.

```
proc summary data=allcardata nway missing;
class carcat;
var ytd current bought sold prev;
output out=national(drop=_type_ _freq_) sum=;
run;

proc summary data=allcardata nway missing;
class region carcat;
var ytd current bought sold prev;
output out=region(drop=_type_ _freq_) sum=;
run;

proc summary data=allcardata nway missing;
class region state carcat;
var ytd current bought sold prev;
output out=states(drop=_type_ _freq_) sum=;
run;

proc summary data=allcardata nway missing;
class region state cities carcat;
var ytd current bought sold prev;
output out=cities(drop=_type_ _freq_) sum=;
run;
```

FORMAT DATA IN XML

XML has very strict rules to rendering and if your output is in any way incomplete it will not import the data into your document. Luckily you can easily validate your XML as you are developing by merely opening it in a browser.

The code to generate the XML from each of the summarized datasets above can be done in a number of ways. You can use the XML engine, or a data null to generate the ASCII file. The example below is a simplified version of the code required, but gives an idea of the process involved.

```
filename regout "pacific.xml";

data _null_;
  set states;
  where region='PACIFIC';
  if n=1 then do;
    put '<?xml version="1.0" encoding="ISO-8859-1" ?> ';
    put '<CAR_COUNTS>';
    put '<ITEM>';
    put '<DESC>' @;
  end;
  put state '</DESC>';
  if carcat='ALL' then do;
    put '<ALLNEW>' ytd '</ALLNEW>';
    put '<ALLCURRENT>' current '</ALLCURRENT>';
    put '<ALLADDS>' bought '</ALLADDS>';
    put '<ALLTERMS>' sold '</ALLTERMS>';
    put '<ALLPREVEFF>' prev '</ALLPREVEFF>';
  end;
  put '</ITEM>';
  if eof then do;
    put '</CAR_COUNTS>';
  end;
run;
```

The first line in each XML File requires a header line identifying it as an XML file with the XML version and encoding type. For example the pacific.xml (for only the Arizona content) will look something like this:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<CAR_COUNTS>
<ITEM>
<DESC>Arizona</DESC>
<ALLNEW>2,198</ALLNEW>
<ALLCURRENT>135,526</ALLCURRENT>
<ALLADDS>1,201</ALLADDS>
<ALLTERMS>477</ALLTERMS>
<ALLPREVEFF>134,802</ALLPREVEFF>
</ITEM>
</CAR_COUNTS>
```

You will notice that each data item has a defining beginning tag and an end tag. This is crucial if the XML is to render properly.

HTML CODE

The web page now has to define the table to be presented and incorporate the XML that has been developed for each level of data.

In the HEAD section of the HTML code you identify the XML source to be used in building the table.

```
<xml id="detail" src="pacific.xml"></xml>
```

In the BODY section of the HTML code a Table is defined and references the id assigned to the XML Source.

```
<BODY>
<TABLE DATASRC="#detail">
```

Each row containing data defines a data field for each cell represented in the table.

```
<tr>
<td><div datafld="DESC"/></td>
<td><div datafld="ALLNEW"/></td>
<td><div datafld="ALLCURRENT"/></td>
<td><div datafld="ALLADDS"/></td>
<td><div datafld="ALLTERMS"/></td>
<td><div datafld="ALLPREVEFF"/></td>
</tr>
```

The following basic HTML code renders the table below

```

<html>
<head>
<TITLE>WUSS 13 Demo</TITLE>
<xml id="detail" src="pacific.xml"></xml>
</HEAD>
<BODY>
<TABLE DATASRC="#detail" border=1 cellspacing=0>
<tr>
<th>&nbsp;</th>
<th colspan="6" name="all">All</DIV></th> </tr>
</tr>
<tr>
<th>State</th>
<th>New Cars<br>YTD</th>
<th>Current<br>on the Road</th>
<th>Bought</th>
<th>Sold</th>
<th>Previous<br>Effective</th>
</tr>
<tr>
<td><div datafld="DESC"/></td>
<td><div datafld="ALLNEW"/></td>
<td><div datafld="ALLCURRENT"/></td>
<td><div datafld="ALLADDS"/></td>
<td><div datafld="ALLTERMS"/></td>
<td><div datafld="ALLPREVEFF"/></td>
</tr>
</TABLE>
</body>
</html>

```

	All					
State	New Cars YTD	Current on the Road	Bought	Sold	Previous Effective	New Cars YTD
Arizona	2,198	135,526	1,201	477	134,802	2,198

CONCLUSION

The amount of planning involved for generating XML driven web pages is crucial to the success of the data delivery. Code in this paper is rudimentary and merely outlines the basics required for simple data presentation. Refer to the presentation and the web page below for more detail regarding navigation through levels of data.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
 Bruce Kayton
 San Diego, CA 92128
 Phone: (858) 335-2138
 Email: bksolutions@the-k-zone.com
 Web: http://www.the-k-zone.com/WUSS_13/

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.