



Science For A Better Life



Exploring Newer Techniques in SAS to Enhance Interactive Web Applications

PhUSE Conference 2016, Barcelona

October 11, 2016 / Volker Harm



Agenda

- Introduction
- The Techniques Presented
- The Techniques Applied
- Recommended Reading



Introduction

- Interactive web applications with SAS
 - Why?
 - How?

Interactive web applications with SAS

Why?



- “From paper to screen”, “Going digital”
 - Our traditional way of working is listing and table oriented
 - sometimes enhanced with some graphics
 - Quite inefficient for exploring and reviewing data
 - Lots of new data visualization solutions
 - Spotfire, Shiny (R package)
 - Once adapted, they have amazing capabilities
 - But high efforts are needed to change the environment as most of our data are SAS data

Explore, if there are alternative solutions using SAS data directly

Interactive web applications with SAS

How?



- Creating html output with SAS
 - Presenting SAS data sets on the Web, PhUSE 2014 using
 - SAS ODS
 - JavaScript and JavaScript libraries as jQuery, tableSorter, dataTables, ...
- Delivering SAS html output
 - Distributing them via email
 - Building web sites integrating SAS html output using a webserver
 - Access data interactively using SAS Stored Processes and the SAS Stored Process Web Application (SAS BI Server)

For all these aspect SAS 9.4 brings major improvements.



The Techniques Presented

- `proc stream`
- `function dosubl`
- `proc json`
- ODS Destination `html5`

proc stream

Motivation



Creating web pages with SAS

- SAS ODS
 - SAS results, tabular or graphical, can be output via ODS as valid web pages.
 - ODS also provides possibilities to integrate JavaScript libraries
- file/put
 - Customized content such as forms must use a data _null_ step and put statements.
 - And this becomes quite fast very cumbersome as HTML uses a lot of double quotes and the puts make code unreadable.

proc stream tries to cure that

proc stream

Features



- enables to process an input stream that consists of arbitrary text
- text blocks may be included (%include)
 - where text may contain SAS macro specifications
 - macro code and specifications will be expanded
- text blocks may be read (readfile)
 - where text is read as it is
- provides special parameters
 - to deal with SAS comments (noabsscmnt)
 - to preserve the columns of the original input file (prescol)
 - how to handle quoting (quoting)
- results can be stored in a file

A lot of things to consider,
but your blog presenting SAS data was never easier to create!

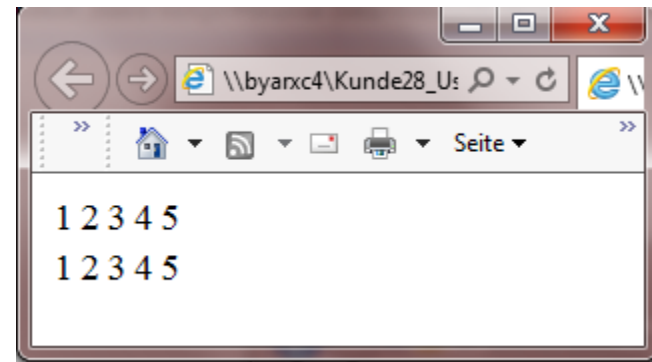
proc stream

Example



```
%macro doit(nr,nc);  
  <table>  
    %do i=1 %to &nr;  
      <tr>  
        %do j=1 %to &nc;  
          <td>&j</td>  
        %end;  
      </tr>  
    %end;  
  </table>  
  
%mend;  
  
filename myfile temp;  
proc stream outfile=myfile;  
begin  
  %doit(2,5)  
  ....  
  ;;;
```

```
<table>  
  <tr>  
    <td>1</td> <td>2</td>  
    <td>3</td> <td>4</td> <td>5</td>  
  </tr>  
  <tr>  
    <td>1</td> <td>2</td>  
    <td>3</td> <td>4</td> <td>5</td>  
  </tr>  
</table>
```



function dosubl

General



- The dosubl function enables the immediate execution of SAS code after a text string is passed.
- Macro variables that are created or updated during the execution of the submitted code are exported back to the calling environment.

Combined with proc stream this allows a data driven construction of web pages.

function dosubl

Example



```
filename _webout "&root/output/IncludingSASCode.html";

proc stream outfile=_webout sqac;

BEGIN

%let rc =%sysfunc (

  dosubl (

    proc sql noprint;

      select distinct  '<input type="checkbox" name="Sex" value="'||strip(Sex)||"'>'||strip(Sex)

      into:checkboxes separated by '<br>' from sashelp.class;

    quit;));

<html><body>

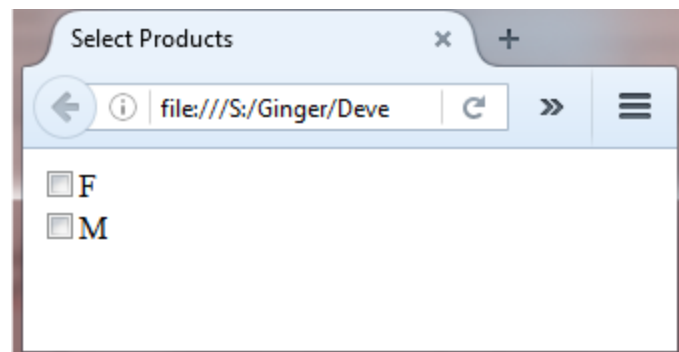
<title>Select Products</title>

&checkboxes

</body></html>

;;;;

run;
```



proc json



- Easily produces JSON from SAS datasets
- using export table
- Can produce more complex JSON as well as simple JSON
- using write open [array or object]
- Can use macros to write PROC JSON, making it very flexible

JSON is the data format of the browser.

proc json cont.



Minimum code needed

```
proc json out=temp ;  
    export sashelp.class ;  
run ;
```

produces default output

```
{  
  "SASJSONExport": "1.0",  
  "SASTableData+CLASS": [  
    {  
      "Name": "Alfred",  
      "Sex": "M",  
      "Age": 14,  
      "Height": 69,  
      "Weight": 112.5  
    },  
    {  
      "Name": "Alice",  
      "Sex": " F",  
      "Age": 13,  
      "Height": 56,  
      "Weight": 85  
    }  
  ]  
}
```

proc json

cont.



Better PROC JSON code

- Pretty - lays out the JSON clearly
- noSAStags - leaves out SAS metadata from the JSON

```
proc json out='/tmp/class.json' pretty  
nosastags ;  
    export sashelp.class ;  
run ;
```

```
[  
{  
  "Name": "Alfred",  
  "Sex": "M",  
  "Age": 14,  
  "Height": 69,  
  "Weight": 112.5  
},},
```

ods html5

- The ODS HTML5 statement creates HTML5 output
 - HTML5 has new additional document structure elements.
 - HTML5 has excellent support for video and audio.

Most important: ODS HTML5 creates scaleable vector graphics.



HTML5: Scalable Vector Graphics (SVG)

- Scalable Vector Graphics is an XML language for describing two-dimensional vector graphics.
- SVG is the default file type for the HTML5 destination.
- SAS can create SVG documents by using Universal Printers and SAS/GRAPH device drivers.
- Most often in SAS, the SVG Universal Printers and device drivers are used to create graphs.
- Graphs can be created by using ODS Graphics or SAS/GRAPH.
- SVG is the default Universal Printer and device driver for the ODS HTML5 destination.
- SVG documents can be stand-alone files or integrated within an HTML5 file.

HTML5: Scalable Vector Graphics (SVG)

cont.



To view an SVG document in an HTML file

- you either create a link to the SVG document,
- embed the SVG document in the HTML file,
- or create an SVG graph that is integrated in the HTML.

SVG mode

- The default value for option `SVG_MODE` for the HTML5 destination is `INLINE`.
- In order to embed the SVG graph, you must specify `OPTIONS (SVG_MODE="EMBED")` in the ODS HTML5 statement.

This ensures that the recipient of an email with html-attachment does not need to care for the graphics files. And the graphics are shown even on smartphones.

The Techniques Applied

- General Structure of Interactive Web Applications Using SAS
- Integration of Visualization Tools Available for Modern Web Browsers
- Drill-down from summary to detailed data
- Switching from graphical to tabular graphical views

General Structure of Interactive Web Applications Using SAS



Components

- SAS Webserver (or other like XAMPP, WAMP, ...)
 - Static content
 - File system
 - in SAS 9.4: ...\\Lev1\\Web\\ WebServer\\htdocs as root for your websites
 - `http://<server:port>/<website>`
- SAS Stored Process Web Application
 - Dynamic content
 - SAS Stored Process Server
 - `http://<server:port>/SASStoredProcess/do?<SAS Stored Process and parameters>`



Webpage from the web server

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

    <title>Search Client</title>

    <link rel="stylesheet" type="text/css" href="/css/AJAX.css">

    <script type="text/javascript" src="/scripts/search_suggest.js"></script>

  </head>

  <body>

    <h2>Search Suggest Demo</h2>

    <form id="frmSearch" method="post" action="http://by-xa221.de.bayer.cnb/SASStoredProcess/do" onsubmit="selectList();">

      <input type="hidden" name="_program" value="/_BI/content/STA_StatApps/Development/WebCOPE/View results">

      <input type="hidden" name="_debug" value="log">

      <div id="optGrp">

        <h5>Datasets to Search</h5>

        <input type="radio" name="table_id" value="FIRMS" checked="checked" onclick="clearIt();" />Pharmaceutical
Firms<br/>

        <input type="radio" name="table_id" value="LISTINGS" onclick="clearIt();" />Pharmaceutical Listings<br/>

        <h5>Search Suggest Method</h5>

      </div>

    </form>

  </body>

</html>
```

Web interface and call of SAS Stored Process



```
%let STPToBeExecuted = /_BI/content/STA_StatApps/Development/WebCOPE/Analyze
data set;

data _null_;
  format infile $char256.;
  input;
  infile = resolve(_infile_);
  file _webout;
  put infile;
  cards4;
<HTML>
  <BODY>
    <H1>Data selection</H1>
    This sample illustrates the data analysis capabilities of stored
    processes.

    <form action = "&_url." method="post" enctype="multipart/form-data">
      <input type="hidden" name="_program" value = "&STPToBeExecuted.">
      <HR>
        <pre><h3>Data source</h3></pre>
        <table border = "1" cellpadding = "5">
          <tr>
            <th align = "left">Analysis Data Set:</th>
            <td><input type = "file" required name =
"AnalysisDataSet"></td>
          </tr>
        </table>
        <HR>
        <INPUT TYPE="SUBMIT" VALUE="Run Procedure">
        <INPUT TYPE="CHECKBOX" NAME="_debug" VALUE="log">Show SAS Log
      </FORM>
    </BODY>
  </HTML>
  ;;;
run;
```

Called stored process

```
* stpAnalyseDataSet;
* uses the tableEditor tagset to present the SAS data set selected by the
calling SAS Stored Process;
%let System = WebCOPE;
%let Version = 0.1;

%include
"F:\Statdb\Ginger\Development\System\Scratch\TableEditor\tableEditor.tpl";

*ProcessBody;
* output option and options for the tableEditor are set in hidden prompts of
the SAS Stored Process;
%global
    _odsstyle
    _odsdest
    _gopt_device
    _debug
    _odsoptions;

%stpbegin;
options nofmterr validvarname = any;

* start log;
%put "&System. &Version.: Run on &sysdate. &sysmtime. by user &_USERNAME.";

* set status message;
%let _status_message = STATUS: &_WEBIN_FILENAME. uploaded;

data AnalysisDataset;
    set &_WEBIN_SASNAME.;
run;

proc print data = AnalysisDataset;
run;

%stpend;
```

Switching from graphical to tabular

```
data class;

    set sashelp.class;

    length urllink $30.;

    if upcase(Sex) = "F" then
        urllink = "href = FemalesUnivariate.html";
    else if upcase(Sex) = "M" then urllink = "href = MalesUnivariate.html";

run;

proc gchart data=Class;

    title1 "Drill-down Graphical Application";
    title2 "Click on a slice";

    footnote;

    pie Sex / html=urllink;

    where upcase(Sex) in ("M", "F");

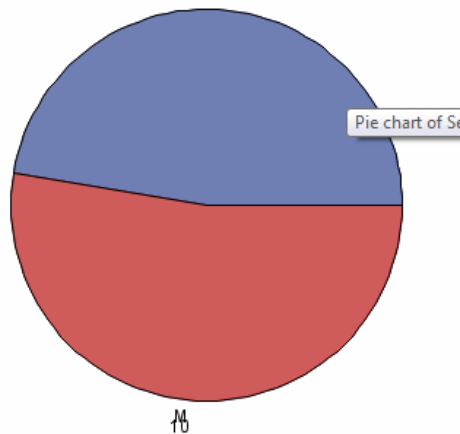
run;
```

Switching from graphical to tabular

Drill-down Graphical Application

Click on a slice

FREQUENCY of Sex
5



Creating a Drill-down for Males

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Age	10	13.4000000	1.6465452	11.0000000	16.0000000
Height	10	63.9100000	4.9379370	57.3000000	72.0000000
Weight	10	108.9500000	22.7271864	83.0000000	150.0000000

[Display Graph](#)

Drill-down from Summary to Detailed

```
proc format;
  value $link
    'M' = 'Males.html'
    'F' = 'Females.html'
  ;
run;

ods html5 path= "&ResultsDir." (url=none) file= 'ClassMeans.html5';

proc report data=sashelp.class nowd;
  title 'Summary of class by gender ';
  title2 'click on a link to see the detail data';
  footnote;
  column sex n age height;
  define sex / group style = {url=$link.};
  define n / 'Count';
  define age / mean 'Average Age';
  define height / mean 'Average Height';
run;

ods html5 path= "&ResultsDir." (url=none) file='Females.html5';

proc print data=sashelp.class noobs;
  title 'Data on Women';
  title2 link = 'ClassMeans.html5' 'Go Back To Summary';
  footnote link = 'Males.html5' 'Detail Report for Men';
  where sex = 'F';
  var name age height;
run;
```

Drill-down from Summary to Detailed

Summary of class by gender
click on a link to see the detail data

Sex	Count	Average Age	Average Height
E	9	13.222222	60.588889
M	10	13.4	63.91

Data on Men
[Go Back To Summary](#)

Name	Age	Height
Alfred	14	69.0
Henry	14	63.5
James	12	57.3
Jeffrey	13	62.5
John	12	59.0
Philip	16	72.0
Robert	12	64.8
Ronald	15	67.0
Thomas	11	57.5
William	15	66.5

[Detail Report for Women](#)



Recommended Reading

- Henderson, Don. 2013. *SAS Server Pages: Generating Dynamic Content*. Cary, NC: SAS Institute Inc.
- Freeman, Eric, Robson Elisabeth. 2011. *Head First HTML5 Programming*. Sebastopol, CA: O'Reilly



Science For A Better Life



Thank you!



11.69

4.38

3.66

0.36

0.36

3.66

4.38

11.69

Content area and guides



5.03

Please restrict your content to this area

0.80

1.77

7.60