Adventures in Building Web Applications: A Tutorial on Techniques for Real-World Applications

Presented at NESUG 2000, Philadelphia
on September 25, 2000
by Jason Gordon, GE Card Services,
Michael Davis, Bassett Consulting Services,
and George Sharrard, GPS Corporation

Presentation Goals

- explain how SAS/IntrNet® operates differently than one might think
- share techniques for automating site updates without using htmSQL
- validating user input without JavaScript
- log usage without hit counters or server logs
- returning SAS/GRAPH output to browser

Purpose of Dashboard Application

- standardize porfolio reporting and measures for authorization and credit line strategies
- give analysts a simple way to obtain portfolio information and diagnostics
- facilitate sharing across multiple locations
- fast way to get and distribute custom reports and graphics for analysis and presentations

Brief Demo

- show "look and flow" of the application
- generate a few sample charts and reports
- contrast appearance of GIF verses PostScript

Developing Web Applications

- Dashboard prototypes developed on PC
- web server and SAS application server hosted on same CPU
- used MS Personal Web Manager and MS FrontPage on Windows NT
- be aware of platform differences such as filename case and *htm* vs *html* filename extensions

Testing Web Applications

- use Hummingbird Exceed FTP to move web site from PC to test directory on production server
- run test pages via launch service to avoid "crashing the socket"
- promote tested applications to production directory via FTP
- segregating production and test pages allow testing revisions to applications before putting into production

Automated String Replacement: Why Is It Needed?

Consider:

```
<form action=
"http://hounddog/scripts/brokerv6.exe"
method="get" target= "_blank">

<input type=hidden name="_service“
value=“newsas_v6”>
```

Automated String Replacement: Example Replacement Strings

Compare with:

```
<form action=
"http://sunweb01/cgi-bin/brokerv6"
method="get" target= "_blank">

<input type=hidden name="_service“
value=“default”>
```

Automated String Replacement: How The Authors Do It

- over one hundred measures get launched by their own web page
- authors used a SAS solution to solve this problem – an SCL global replacement program
- use SCL functions to identify HTML directory entries and to read them as data files
- why SAS? portability between Windows and Unix

Automated String Replacement : G_RPLACE.SCL

- control file sets which strings to replace
- search strings in SCL lists for performance
- assumes that “find string” begins line
- use asterisk to suspend replacement
- good model for that FAQ on SAS-L
- GIF and JPEG can be copied using SASHELP.FSP.IMGDAT Class

Automated Block Replacement: Why Not htmSQL?

- authors’s technique appropriate when selections change infrequently
- primary changes required include drop-down lists and radio station selections
- *What’s New* in the Help system changes
- why incur overhead, expense of htmSQL when a page only changes monthly ?

Automated Block Replacement: Blocks and Blocks Tags

- block start and block end tags are embedded the HTML pages (*see example*)
- tags are HTML comments and are ignored
- before B_RPLACE.SCL is run, the replacement blocks are revised by hand or maintenance program
- program replaces HTML code between tags

Automated Block Replacement: B_RPLACE.SCL Control Table

- table links start and end tags with the filename of the replacement block
- additional columns further restrict which category of HTML pages are to be updated
- category is set by the page name prefix
- advantages: faster updates, selective updating, single HTML directory

Validating User Input Without JavaScript

- most web developers use JavaScript to validate user input
- authors wished to be able reset user input validation edits programmatically
- easier to change a data set observation than to search and alter an embedded script

Validating User Input: The SCL Validator

- executed through a PROC DISPLAY
- macro variable *err_flag* is zero (0) if all check pass
- macro variable *err_msg* contains short text message from first failed check
- program that executes the Validator is responsible for interpreting macro vars from HTML form created by broker

Validating User Input: Validator Control Table

Variable	***Description***	***Example***
chktype	check type	“constant”
chkvar	check variable	“client”
compvar	comparison variable	
comp_op	comparison operator	“ne”
constant	comparison constant	“NoClient”

Validating User Input: Validator Control Table (cont’d)

Variable	***Description***	***Example***
fail_msg	failure or warning msg	
num_char	numeric or character	“C”
severity	severity of comparison	1
valpage	HTML pp. to validate	“pp_”
vformat	variable format	“$28.”

Logging Without Hit Counters or Web Logs

- hit counters measure page visits, not usage
- hit counters are platform specific
- web logs show visits, not selections
- web logs can be intimidating to analyze
- Dashboard application writes out all user selections from the HTML form and broker, along with timedate stamp

Appending the Dashboard Log File

```
proc append
 base=logger.logger data=logger force ;
run ;
```

- *Stupid Dog Trick…* from FTP explorer program, use SAS Viewer to look at Unix files on a desktop computer or paste into MS Excel

Let's Make Some Output: _webout

- fileref automatically allocated by SAS/IntrNet Application Server
- content sent to *_webout* placed in same or new browser window
- in Version 8, use ODS to create HTML
- in Version 6, process is similar to writing reports to a file via a DATA _NULL_ step
- similar but different…

SAS/GRAPH® Drivers

- outside SAS/IntrNet, use WINxxx
- Windows handles the hardware issues
- web application might specify:

```
goptions device= gif ;
```

- problems: low resolution, inability to send multiple graphics directly to *_webout*

Writing the HTML Header

```
data _null_ ;
   file _webout ;
   put 'Content-type: text/html' ;
   put ;  /* must write blank line */
   …
run ;
```

This Does Not Work !

```
goptions device=gif gsfname=_webout
gsfmode=append rotate=landscape display ;

%include "path2./replay1.sas" ;

goptions device=gif gsfname=_webout
gsfmode=append rotate=landscape display ;

%include "path2./replay2.sas" ;
```

Embed Graphics With This Technique

```
filename graf01ga "<path>/&fiz.a.gif" ;
goptions device=gif gsfname=graf01ga
gsfmode=append rotate=landscape display ;

%include "<path>/replay03a.sas" ;

data _null_ ;
   put …
   "<img src= http://<url>/&fiz.a.gif"> ;
```

Fonts

- GIF and JPEG created on a Unix Application Server do not have access to hardware fonts, such as True Type
- result is small text and “pixelation”
- one solution: generate second optional image in PostScript
- user clicks on link to PostScript file

Version 8

- Dashboard application was built in Ver. 6 because original web server OS was not capable of supporting Version 8
- current web server supports both versions
- plan include PDF, hardware font
- Pool Service to support higher demands
- tip: `ods html body=_webout (dynamic) ;`

Acknowledgements