Paper 183

The Web Enablement of the SESUG 97 Registration Process and Other Applications: Perl or the SAS® Application Dispatcher?

Greg Ashley, University of Georgia, Athens, GA

Abstract

The 1997 SouthEast SAS® User's Group's (SESUG) conference was the first SESUG conference to offer Web-based registration. This paper examines how the *SAS® Output Formatter* and various *CGI tools* were utilized in the Web-based online registration process for which the author served as registrar.

It then compares and contrasts these tools to the use of the SAS® Application Dispatcher which could alternatively be used for future SESUG conference registrations or other Web-based data applications.

Additional uses of the SAS® Application Dispatcher are planned at the University of Georgia and some of these possible applications are also presented.

Introduction

The SESUG '97 online registration process used *HTML forms, perl, sendmail, procmail, and the SAS® HTML Output Formatter* in a convoluted behind-the-scenes process to gather registration data from conference attendees, process this data, and then build a database of attendees and associated information. Although this method was functional it could certainly be simplified.

One alternative to this process would be to use *HTML forms* and the *SAS® Application Dispatcher*. The *SAS® Application Dispatcher* lets you create a Web user interface and collect and process data without having to write your own CGI scripts in perl or similar languages – instead you can use SAS®.

Background: Web Enablement with SAS®

SAS® has many tools for Web Enablement and two of these are examined in detail in this paper:

- The HTML Output Formatter
- The SAS® Application Dispatcher.

The HTML Output Formatter lets you create an HTML file that contains the output from SAS® procedures. The Output Formatter creates your SAS® output with valid HTML tags. The resulting file doesn't just contain output surrounded by <PRE> tags, but rather it contains your SAS® output formatted using custom HTML tags.

The Application Dispatcher is a SAS®/IntrNet component and functions as a Web gateway from your Web browser to SAS®. This gateway, written using the Common Gateway Interface (CGI), provides access to data in combination with SAS® analysis and presentation procedures. Additionally, SAS® software does not have to be installed on the user's machine.

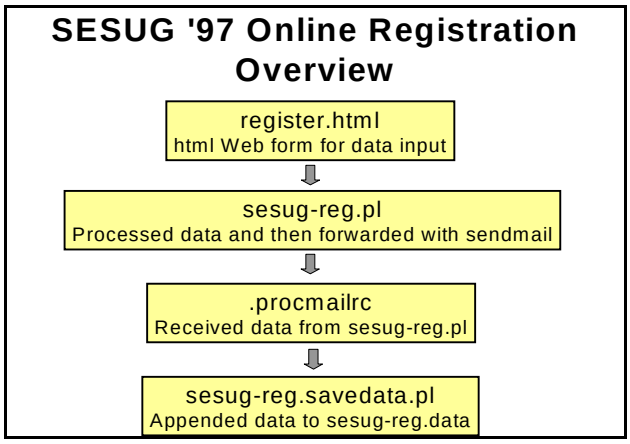
The SAS® Application Dispatcher works in conjunction with HTML forms where users select items and fill in fields. When a user submits the information, the Dispatcher passes the information to the CGI program and on to a waiting SAS® session. SAS® software processes the information using a SAS® program that is identified in the HTML form. Program results return through the CGI to the browser and display on the screen.

CGI programming experience is not required in order to use the Application Dispatcher and you can create the Web user interface and retrieve SAS® data for display on the Web without having to program a CGI script in something like perl.

Complete details of all of the SAS® Web enablement tools can be found at the SAS® Web site at www.sas.com.

An Overview of the 1997 SESUG Web-based Registration Process

SESUG '97 attendees entered registration information via a HTML form (register.html) which passed the data to a perl script (sesug-reg.pl) whose primary function was to use sendmail to forward the data to the official SESUG account with the subject "sesug-reg-webbased". The .procmailrc file for the official SESUG account routed any message that arrived with the subject of "sesug-reg-webbased" to another perl script (sesug-reg.savedata.pl) that appended the record to a file (sesug-reg.data).

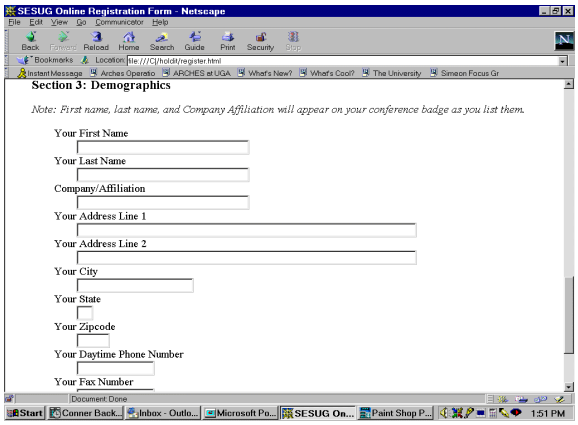


Excerpts from *register.html*

This was just plain HTML that collected the registrant's information and then invoked the perl script *sesug-reg.pl...*

```
<FORM method=POST
ACTION="http://www.uga.edu/sesug-bin/sesug-
reg.pl">

<stuff deleted here>

<DD><INPUT NAME = "fname" SIZE="30">
<DT>Your Last Name
<DD><INPUT NAME = "lname" SIZE="30">
<DT>Company/Affiliation
<DD><INPUT NAME = "company" SIZE="30">
<DT>Your Address Line 1

<more stuff deleted here>
```

The Registration Web Form: *register.html*



Excerpt from *sesug-reg.pl*

This is a perl script that processed the data received from register.html and used sendmail to e-mail it with the subject "*sesug-reg-webbased*" to the official SESUG computer account...

```
<stuff deleted>

       open(SENDMAIL,"|/usr/lib/sendmail -t");
print SENDMAIL "To: sesug\@www.uga.edu\n";
print SENDMAIL "From: www\@www.uga.edu\n";
print SENDMAIL "Subject: sesug-reg-webbased\n";

<stuff deleted>

print SENDMAIL "$string\n";
close(SENDMAIL);
```

The *.procmailrc* file for the official SESUG e-mail account

Any e-mail that arrived in the SESUG account with the subject "*sesug-reg-webbased*" was passed to the perl script *sesug-reg.savedata.pl* for processing...

```
PMDIR=$HOME
LOGFILE=$PMDIR/procmail.log

:0
* Subject:.*sesug-reg-webbased*
| /usr/www/sesug/bin/sesug-
reg.savedata.pl

:0
*
! gashley@arches.uga.edu
```

Excerpt from *sesug-reg.savedata.pl*

This perl script appended the registration data to the file: *sesug-reg.data...*

```
<stuff deleted>

while (<STDIN>) {
     last if ($_ eq "\n");
}
open(REGDATA,">>/usr/www/sesug/sesug-reg.data")
||
  die "can't open data file";
while (<STDIN>) {
    print REGDATA $_;
}
close(REGDATA);
```

Data Backups

The file *sesug-reg.data* was backed up several times every a day via a cron job. A unique time and date extension was added to each backup file and it was written to another remote system for added security. For example, *'sesug-reg.data.09-18-12'* would indicate that this was the September the 18th noon backup file.

Publishing Reports From the Registration Data:

A SAS® program was executed periodically on the author's desktop PC that read data from one remote system and wrote HTML formatted output to another remote system using the ftp option on the filename statement. These reports were used by conference planners.

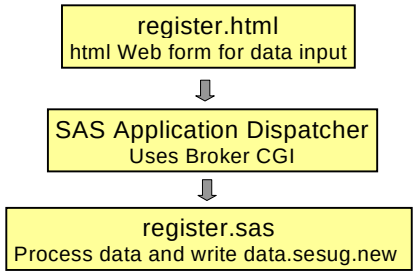
1. Data was read remotely from the most recent backup file (e.g. sesug-reg.data.09-18-12, etc.) using the ftp option on a filename statement

2. The SAS® HTML Output Formatter was then used to generate a Web page report from this data and publish it back to the official SESUG account where conference planners could browse it.

Excerpts of the SAS® code that was executed on the author's desktop PC which read registration data and wrote HTML reports on remote systems follows:

```
data one;

* prepares to read data from the most recent
backup;

filename sesugin ftp 'sesug-reg.data.09-18-12'
 user='gashley' host='sb.dcs.uga.edu'
 recfm=v pass='xxxxxx';

* html to be written to another system after
processing;

filename sesugout ftp 'sesug-short.html'
cd='/sesug/public_html'
 user='sesug' host='www.uga.edu'
 recfm=v pass='xxxxxx';

data sesug97; infile sesugin delimiter=",";

length fname $ 30; length lname $ 30;

input fname $ lname $ guest1 $ guest2 $ email $
phone $ fax $ address1 $ address2 $ city $
company $ state $ zip $ wkshp_am $ wkshp_pm $
lunch $;

proc sort; by lname;   run;

* SAS HTML output formatter to generate a HTML
report begins here;

%out2htm(capture=on,
         window=output,
         runmode=b);

proc print noobs uniform; var lname fname;

%out2htm(htmlfref=sesugout,
        capture=off,
        window=output,
        openmode=replace,
        runmode=b,
        bgtype=color,
        bg=#ffffff);
```

The Alternative Online Registration Method Using the SAS Application Dispatcher



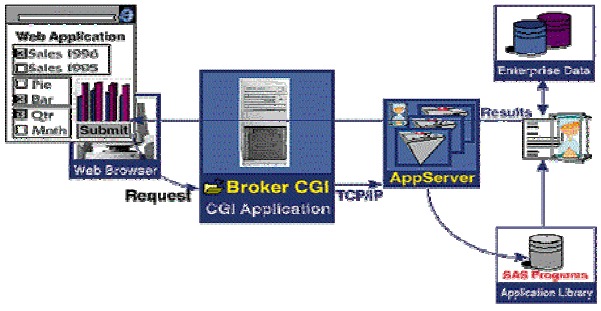
How the Application Dispatcher Works

1. A user fills out fields in an HTML form from their Web browser and submits it. The information from the form is passed to the Web server, which invokes the first component of the Dispatcher called the Application Broker.

2. The Broker retrieves the data and sends it to the second Dispatcher component, called the Application Server. Information passed to the Broker tells the Broker which Server should process the request.

3. The Application Server invokes a SAS® program that processes the information.

4. The results of the program are streamed back through the Broker to the browser.



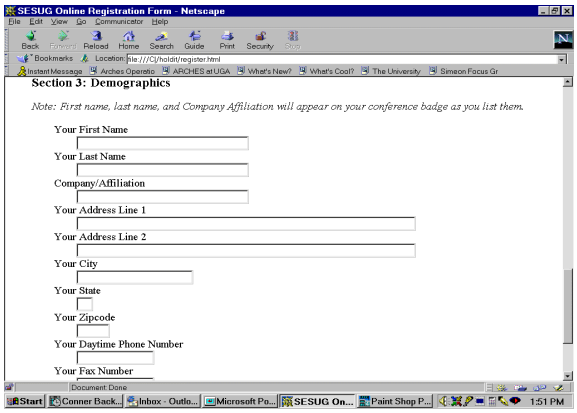
(source www.sas.com/rnd/web/dispatch/how.html)

Excerpts from the Alternative *register.html*

This form is almost identical to the '97 *register.html*, except that this one invokes the SAS® Application Dispatcher via the "*broker cgi*" script and passes the data as SAS® macro variables to the SAS® program *register.sas,* the remainder of the HTML form is the same as the one for SESUG '97 ...

```
<FORM ACTION="http://sb.dcs.uga.edu/cgi-
bin/broker"
 onSubmit="return validate()">

<stuff deleted>

<INPUT TYPE="HIDDEN" NAME="_PROGRAM"
     VALUE="register.sas">
<INPUT TYPE="HIDDEN" NAME="_SERVICE"
     VALUE="default">
<DD><INPUT NAME = "fname" SIZE="30">
<DT>Your Last Name

<stuff deleted>
```

Alternative *register.html*



Getting the data from the Web into a SAS® data set

Register.sas is just a simple SAS® program that opens an existing SAS® dataset containing hypothetical registration data. Next, it appends the data passed from the SAS® Application Dispatcher to this dataset and finally it returns a message stating that the user is registered.

Excerpts from *register.sas*

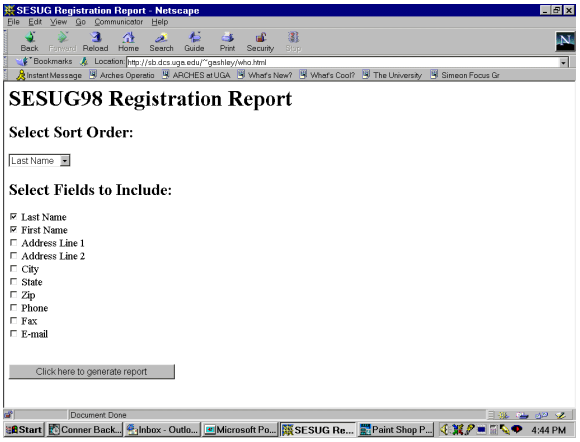
```
* get existing registration data;
  data one; set sample.sesug;
  run;

* extract the data from the macro variables that
were passed from the Application Dispatcher
Broker CGI;

data two;
fname=symget('fname');
lname=symget('lname');

* append the new data to the existing
registration data;

data sample.sesug; set one two; run;

* Send a message back to the registrant to let
them know that they are registered;

data _null_;
  file _webout;

put '<HTML>';
put '<HEAD>';
put'<TITLE>You are registered</TITLE>';
put '</HEAD>';
put '<BODY>';
put '<H2>Your registration information
follows:</H2>';

put "<b>Name (last, first):</b> &lname,
&fname<br>";

put '<br>';

 run;
```

Generating Reports with the SAS® Application Dispatcher for conference planners

The SAS® Application Dispatcher could also be used to query existing data sets to provide reports for conference planners. Below is a simple example, but it could be adapted to display complex queries for diverse data applications. Some benefits that this approach would have over the 1997 SESUG reports are:

- Dynamic - the user selects only the information that they want to see
- Real-Time - the database is updated with each Web form registration submission
- Reports are generated on demand by Conference Planners

Dynamic Real-Time Reports for Conference Coordinators



The HTML Behind the Dynamic Reports (using the SAS® Application Dispatcher)

```
<stuff deleted>

<FORM
ACTION="http://sb.dcs.uga.edu/cgi-bin/broker">
<INPUT TYPE="HIDDEN" NAME="_PROGRAM"
VALUE="report.sas">
<INPUT TYPE="HIDDEN" NAME="_SERVICE"
VALUE="default">

<H2>Select Sort Order:</H2>
<SELECT NAME="order">
<OPTION VALUE="lname">Last Name
<OPTION VALUE="company">Company
<OPTION VALUE="city">City
<OPTION VALUE="state">State
<OPTION VALUE="zip">Zip
</SELECT>
<br><br>
<h2>Select Fields to Include:</h2>
<INPUT TYPE = "checkbox" NAME="lname"
Value="lname" checked="yes">Last Name<BR>

<stuff deleted>

<input type="submit" value="Click here to
generate report">
```

The SAS® code behind the Dynamic Reports (report.sas)

```
<stuff deleted>

data one; set  sample.sesug;

proc sort; by &order;
proc print; var &lname &fname &company &address1
&address2 &city &state
&zip &phone &fax &email;

run;

%out2htm(capture=off,
         runmode=s,
         htmlfref=_webout,
         bgtype=color,
         bg=white,
         brtitle=Means Output);
run;
```

Other Uses for the Application Dispatcher at UGA

We are considering several other uses of the Application Dispatcher at University Computing and Networking Services at UGA. Most notably we are considering implementing the SAS® Application Dispatcher as part of a process to solicit client feedback about "*ARCHES*", which is our campus e-mail system and "*WebCT*", which is our Web-based instructional application. As of January 1999 there were over 30,000 accounts on each system. Naturally, client satisfaction is of paramount concern to those of us that manage and support these systems.

The intent is to create a Web form to solicit client feedback about each system. Randomly selected clients from each system will be periodically invited to complete a client satisfaction survey via the Web.

The objective is to develop a continuum of information for trend and satisfaction analyses. This is will allow us to glean information to help us allocate our limited resources in the manner best suited to provide the systems and services that are in greatest demand.

As with the SESUG application the basic system will consist of a Web-form front end which sends data to the Application Dispatcher for processing. Standard reports will also be available via the Dispatcher.

Given the necessity of conducting a statistically valid sample it is essential to ensure that only those randomly selected clients are able to complete the Web-form. Additionally, it will also be critical to ensure that each respondent only submits the form once. To accomplish this a SAS® data library with the necessary information to be used for verification will likely be maintained.

One major advantage of using the Application Dispatcher is that current staff have significant SAS® expertise and can easily setup and maintain this system. Additionally, it is highly desirable to have the power of the SAS® system to generate the numerous reports and analyses that will be necessary for management.

Summary and Conclusion

The SESUG '97 Web-based registration was well received by registrants, but the convoluted behind-the-scenes process left much room for improvement. The SAS® applications Dispatcher offers an excellent alternative to this process and should be considered for many other Web-based data applications. We are also looking forward to using the Application Dispatcher with our other pending projects. The SAS® Application Dispatcher is very functional and easy to use and if you are experienced with SAS® then there is little or no learning curve. You may want to use the SAS® Application Dispatcher if:

- You have SAS® programming experience but have little or no CGI programming experience. Dispatcher lets you create the Web user interface and retrieve the SAS® data for display on the Web without having to write a CGI script.

- You want to create applications that provide Web output without investing a lot of programming or learning time.
- You want to analyze and display information dynamically on the Web, letting your Web browser users retrieve the information they need immediately.
- You want to create applications that run on a variety of browsers.

Acknowledgments

The author wishes to express thanks to Greg Whitlock for the substantial perl and procmail programming efforts and to Mark Plaksin for installing and configuring the SAS® Application Dispatcher.

About this Paper and the Author

Greg Ashley is the Manager of Academic Applications Support for University Computing and Networking Services at the University of Georgia. He can be reached via e-mail at gashley@uga.edu.