

SAS/IntrNet & Javascript



Alan Maas

Search for client names containing:

Return all clients from:
University of WI at Madison - Research on Poverty - Madison WI
University of WI at Madison - UW DoIT- B332 - Madison WI
University of WI at Milwaukee - Information Media Technologies - Milwaukee WI

Results:

Maas, Alan - <i>Inactive</i>	Scheduled Followup date:09/09/9999
Client Contact Information	
Company: Systems Seminar Consultants	
Address1: 529 w washington st	
Address2:	
Location: Madison , WI 53703	
Department:	
Title: Tester	
Work Phone: ###-###-####	
Extension: .	
Other Phone: ###-###-####	
Fax: ###-###-####	
Email Address:	
Client History	
<input type="button" value="Del"/>	02/21/2006 - JENNIFER FIRST - Inactive - Make inactive
<input type="button" value="Del"/>	02/09/2006 - ALAN MAAS - null - SSC employee validation check. Working.
<input type="button" value="Del"/>	02/09/2006 - ALAN MAAS - Info - Date validation works.

Systems Seminar Consultants

2997 Yarmouth Greenway Drive

Madison, WI 53711

(608)-278-9964

www.sys-seminar.com

Part 1: SAS/Intrnet



SAS/IntrNeT – What is it?

- Server-side (Normally)
- Allows you to set up a web page that sends information to a SAS session
- Greatly simplifies SAS reporting for non-SAS familiar users
- Allows users to use SAS without having it installed on their PC
- Stores and/or retrieves information in SAS data sets
- Can display SAS results in your browser

Note: Setting up SAS/Intrnet requires much more explanation and is somewhat complex. As a developer though, you only need to know how to access the services that are set up for you to use.

Part 1: SAS/Intrnet



Basics of using HTML to kick off SAS/IntrNet - Forms

```
<HTML>
.
.
.
<FORM ACTION='HTTP:\\S4\SCRIPTS\BROKER.EXE' METHOD='POST'>
<INPUT TYPE='SUBMIT' style="width:400" VALUE='Run Program'>
<INPUT TYPE='HIDDEN' NAME='_PROGRAM' VALUE='folder.program_name.sas' _DEBUG=131 >
<INPUT TYPE='HIDDEN' NAME='_DEBUG' VALUE='131'>
<INPUT TYPE='HIDDEN' NAME='_SERVICE' VALUE='SERVICE_NAME'>
<INPUT TYPE='HIDDEN' NAME='_PRIV' VALUE='YES' >
<INPUT TYPE='HIDDEN' NAME='_EXIST' VALUE='NO'>
</FORM>
.
.
.
</HTML>
```

- The `<FORM ACTION="">` points us to our Application Dispatcher.
- Hidden variables are used to instruct the Application Dispatcher what SAS program to run and the values of certain options when the form is submitted.

Part 1: SAS/Intrnet



Basics of using HTML to kick off SAS/IntrNet – Forms continued

A form is denoted by `<FORM>` and has optional variables `name=`, `action=`, `method=` and `target=` to name a few. A form is ended with `</FORM>`

- Every `<INPUT>` variable within the form tags will be sent to the SAS program it points to as macro variables when the form is submitted.
- Different forms can point to different SAS programs.
- Each single form may only point to a single SAS program (Exception using javascript)
- Most reporting interfaces will only require one form

What kinds of things can I send to a SAS program?

Part 1: SAS/Intrnet



Basics of using HTML to kick off SAS/IntrNet – Input variables

Any HTML input variable contained inside the submitted form can be used.

- `<input type="text" name="mytext">`
macro name: mytext, value held: user input into the textbox as a string of characters
- `<select name="mydropdown">` macro name: mydropdown
`<option value=null default>-Select an item-` value held: value of the selected label.
`<option value=12.99>Item1`
`<option value=39.95>Item2`
`<option value="sold out">Item3`
`</select>`
- `<input type="hidden" name="myhidden" value='myvalue'>`
macro name: myhidden, value held: value specified, stored in macro.

Well what possible uses could these HTML input fields have for SAS to the user?

Note: Drop down menus allowing multiple selections and radio or checkboxes create macro variables that are much less intuitively named but using them is by no means difficult.

Part 1: SAS/Intrnet



Basics of using HTML to kick off SAS/IntrNet – Variable purpose

Having macro variables with user defined values at your disposal, you can now create dynamic reports in SAS based on user input.

Examples

- Subset a dataset with where variable=&mytext; before a PROC REPORT
- Perform a PROC PRINT of all transactions with the specified account ID
- Move all records from a dataset meeting certain criteria into an archive
- Use a 'where' clause with the incoming macro variable(s) to speed up searching or reporting

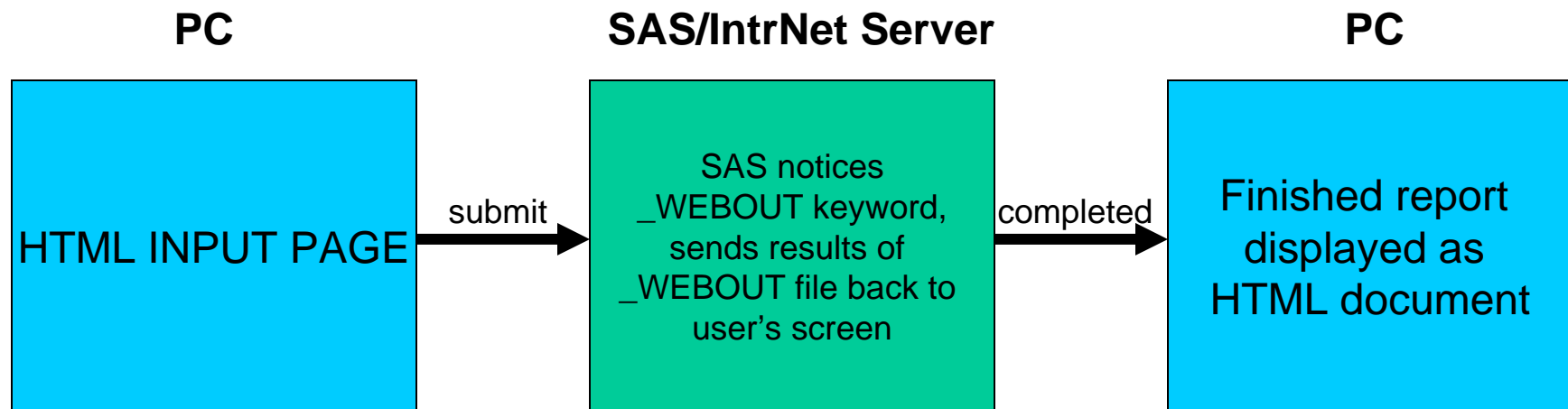
While many of these procedures are basic reporting techniques, allowing the user to view the results of a SAS program started with SAS IntrNet requires something new. `_WEBOUT`

Part 1: SAS/Intrnet



Displaying Results – Basics of _WEBOUT

_WEBOUT is a SAS/IntrNet keyword that tells SAS to send all the output that would normally go to a specified file to go to the user's screen instead. The reports or results must be sent as an HTML file.



Note: If you use a data step and not a specific ODS style, the user's browser will be expecting an HTML file. This will be shown in the examples.

Part 1: SAS/Intrnet



Displaying Results – ODS Method

Program Submitted

```
ODS HTML  
body = _webout;  
  
proc print data=&mydataset;  
title "Proc Print of &mydataset";  
run;  
  
ODS HTML close;
```

Output Produced

SYSTEMS SEMINAR CONSULTANTS, INC.

Save As

PUBLIC ENROLLMENT FOR SASA

Name	ProjectNumber	ClassCode	Company	Phone	EmailAddress
Raymone Jackson	003709	SASA	Northwestern Mutual	414-665-5923	raymonejackson@northwesternmutual.com
Carolyn Lueder	003709	SASA	General Casualty	825-5531	carolyn.lueder@generalcasualty.com
Stephenson, Elicabeth	003709	SASA	DHFS		stephes@dhfs.state.wi.us
Sprangers, Ms. Jeanne	003709	SASA	International Truck and E	262-780-5315	jeanne.sprangers@nav-international.com
Otles, Zekai	004134	SASA	Frontier Science Technology R.	(608)-441-2942	otles@fstfr-wi.org
Nelson, Torrey	004134	SASA	State of WI- DHFS	(608) 267 0235	nelsotp@dhfs.state.wi.us

PUBLIC ENROLLMENT FOR SASA

Frontier Science Technology Res

- Pros:
- Use of ODS significantly simplifies generating HTML reports.
 - All familiar procs used for reporting are available.
- Cons:
- Style and formatting is restricted by SAS where HTML allows more.
 - Complex interactivity such as input choices based on data is impossible.

Note: For this example assume &mydataset was specified in a text box by the user prior to submitting.

Part 1: SAS/Intrnet

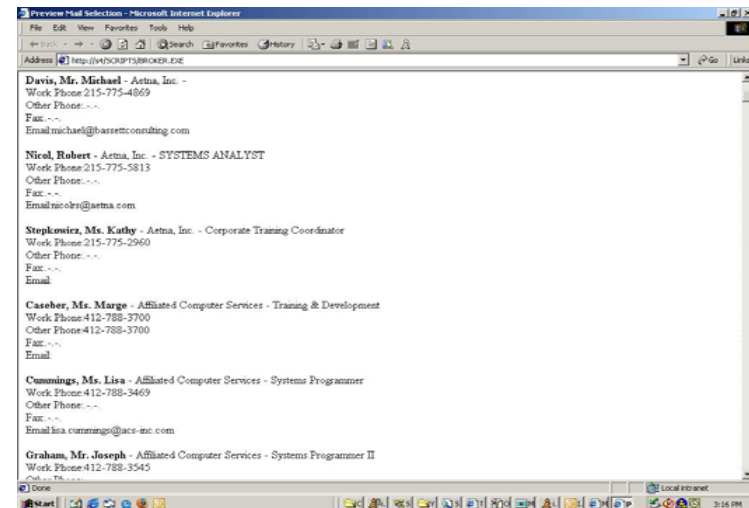


Displaying Results – PUT statement method

Program Submitted

```
DATA _null_;
File _WEBOUT;
PUT '<HTML><HEAD></HEAD>';
PUT '<BODY>'
DO UNTIL (ENDLIST=1)
SET &mydataset end=ENDLIST;
  PUT '<b>Name: </b>' name;
  PUT '<b>Rating: </b>' num_sales;
END;
PUT '</BODY>';
PUT '</HTML>';
```

Output Produced



- Pros:
- Free reign over formatting with html, creating unique reports.
 - Very complex data-based input interaction is possible.
- Cons:
- Extremely tedious; should only be used when necessary.
 - More difficult to debug: two translations.

Part 2: Javascript



Yay! We can create reports!



But what does that have to do with Javascript?

Note: This cute yet terrifying child is not mine.

Part 2: Javascript



What is Javascript?

- Javascript is a cross-platform, object-oriented scripting language that can be interpreted by browsers while embedded in HTML code.

Javascript enhances basic HTML by adding support for...

- instanced variable creation / manipulation.
- user-defined functions, which then can be called during onClick, onMouseOver, onMouseout, onChange and a plethora of other triggers.
- namespace object reference of form input variables.
- built in functions for math, string manipulation etc.
- arrays: both generically and as items in a drop down menu.
- change almost any setting of an existing HTML object.
- contingent creation of new HTML objects.

Note: Javascript is based on Java but the two languages are fundamentally very different.

Part 2: Javascript



Incorporating Javascript into an HTML file

You can use Javascript in a few different ways.

- All code inside `<script>...</script>` will be interpreted as Javascript code. Optionally you can specify `<script language=""></script>` for safety.
- `<script src="path\filename.js"></script>` is equivalent to a %INCLUDE statement in SAS, including and executing all JS code from that file.
- The event handlers (triggers) listed on the prior page automatically assume JS code will follow. Thus `onClick="..JSCODE.."` would execute that code when the event triggered.

Part 2: Javascript



Examples of Javascript code.

1. Initialize values of HTML variables

```
onLoad= "Document.myform.mytextbox.value='Hello World!';  
        Document.myform.mydropdown.selectedIndex=0;  
        ...  
        Document.myform.mytextbox2.value=50;"
```

2. Create functions to display results in a popup window

```
function do_square(my_x)  
{  
    return my_x * my_x;  
}  
alert("The square of 4 is" + do_square(4);)
```

3. Add members to a drop down menu

```
document.form.mydropdown.options[5] = new Option(' Doll House ', '29.95');
```

4. Event handle using 'this' keyword

```
<input type='text' name='firstname' onFocus="this.value='Enter your name';">
```

Part 2: Javascript



Demonstration of how to create dynamic dropdown menus using SAS/IntrNet and Javascript.

Begin Demo