SESUG Paper 198-2019

Integrating SAS/IntrNet, SAS Macro facility, JavaScript, HTML, and .NET to build a dynamic web application to present NSSE data

Carlos Martinez, Wen Jiang, and Uday Nair, University of Central Florida.

ABSTRACT

This presentation is intended for staff in the institutional research, survey research, or assessment office in the education industry who conduct and report survey data. The NSSE institute provides summary and item-level reports comparing a select institution to various other groups of institutions. However, internal university analysis by student characteristics is more insightful and actionable.

A dataset was created that combines NSSE data with the typical institutional research type data. The SAS ODS HTML destination within the SAS/IntrNet ® platform, HTML, and JavaScript were combined to create a dynamic web application consisting of cascading input query pages, custom reporting and easy-to-navigate reporting interface. PROC TABULATE was heavily used to build standard aesthetically-pleasing table templates. The SAS MACRO language was leveraged to improve efficiency of data processing, reporting, and back-end code maintenance. The web application was nestled inside a .NET controlled framework to ensure data access to select users.

This paper does not serve as instructions on how to build such a complex web application from the ground up. Instead, the goal is to highlight some of the more challenging tasks of integration and provide an explanation as to how these were achieved. Knowledge of the programming languages SAS, JavaScript, HTML, and VB.NET or C# will be helpful in understanding the concepts covered in this paper. The presentation will include typical code examples. This paper is organized by certain features in the web application that were achieved through integration of several platforms and languages:

- Creating uniform layout and page sizing [cross-domain integration]
- Dynamically generated cascading dropdown menus
- Printing capabilities from the web application [cross-domain integration]
- Tracking web application user behavior [using Google Analytics]

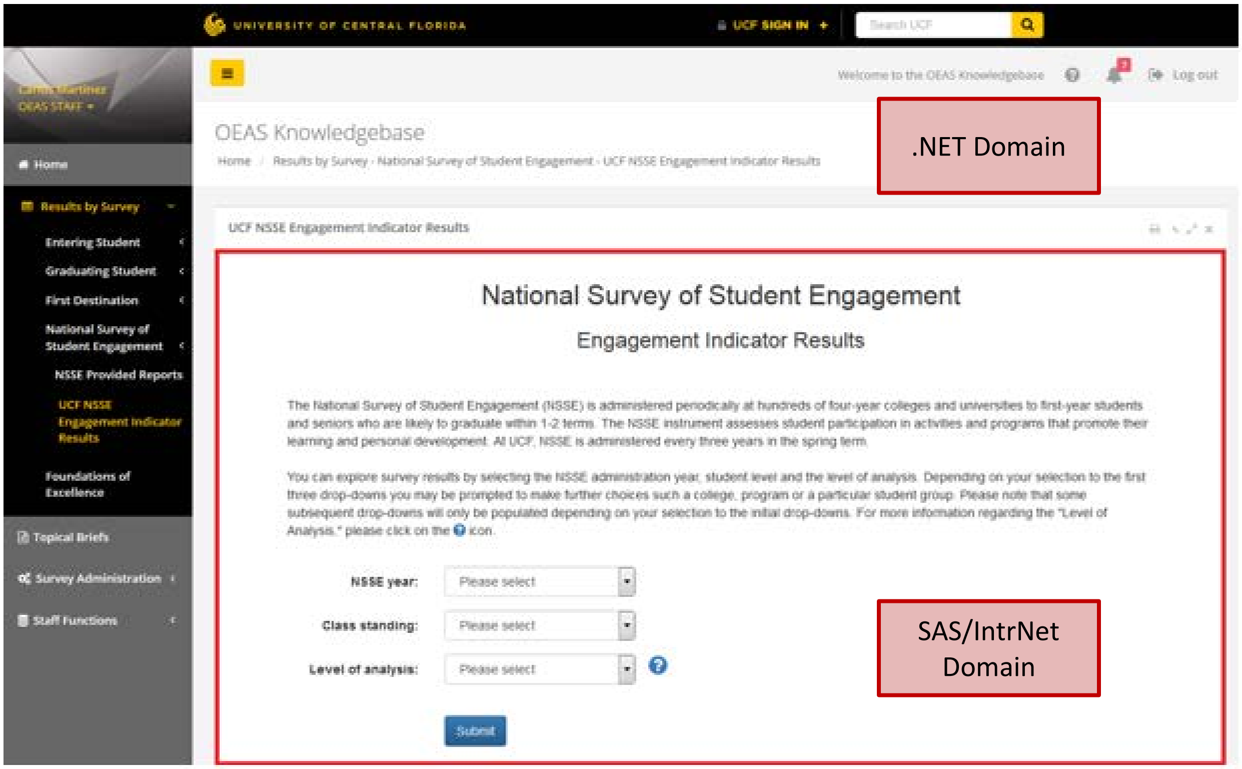
INTRODUCTION

In today's world of information sharing and integration, web applications have become the standard method of distributing data to empower the end-user with information to effectively and efficiently accomplish their objectives. Integrating technologies to develop a fluid dynamic web application can sometimes be a difficult task, especially when developers differ in their background, expertise, and knowledge of programming languages. The University of Central Florida's Operational Excellence and Assessment Support (OEAS) department developed and maintains a web application, OEAS Knowledgebase, to share survey results with internal university faculty and staff. One such survey, the National Survey of Student Engagement (NSSE), is administered at UCF every three years to assess student participation in activities and programs that promote their learning and personal development. The results provide findings about how undergraduates spend their time and what they gain from attending their college or university. Different technologies had to be integrated in order to deliver a seamless web application. It was important to ensure that only the university stakeholders had access to the queries, they could slice & dice the data as needed to effectively retrieve information, and that the web application was easy to navigate. Technologies including SAS/IntrNet, SAS Macro facility, JavaScript, HTML, and .NET, had to be integrated to accomplish these objectives and will be the focus of this paper. This paper does not serve as instructions on how to build such a complex web application from the ground up. There are several resources available on how to get started on your own integrated data reporting systems. Instead, the goal of this paper is to highlight some of the more challenging tasks of integration and provide explanation as to how these were achieved. Knowledge of the programming languages SAS, JavaScript, HTML, and VB.NET or C# will be helpful in understanding the concepts covered in this paper.

OEAS KNOWLEDGEBASE

OEAS Knowledgebase is the web application that houses data collected from over 50,000 students using over 100 survey instruments from over 200 academic programs over the course of the entire academic year. The survey data is collected using the Qualtrics survey platform, administered in some cases within the student portal powered by PeopleSoft, processed using SAS Enterprise Guide ®, and delivered using SAS IntrNet ®, HTML, JavaScript technology, and is housed within a secure .NET framework. It comprises of two domains, https://knowledgebase.ucf.edu and https://sas.oeas.ucf.edu. These domains utilize iframes and security parameters. The security parameters, for the most part, are passed between the domains within the uniform resource locator (URL) query string. The screenshot below shows what a user sees when logged onto the system (Figure 1).

The OEAS Knowledgebase domain, created using .NET, controls user authentication, landing page layout, survey items within the menu on the left-hand side, and basic functions such as printing and pagelet size, etc. The survey data (SAS data sets), the user input query pages, and the SAS programs used to generate output tables from these queries are housed within SAS/IntrNet Domain.


UNIVERSITY OF CENTRAL FLORIDA
UCF SIGN IN
Search UCF
Carlos Martinez
OEAS STAFF
Welcome to the OEAS Knowledgebase
Log out
Home
Results by Survey
Entering Student
Graduating Student
First Destination
National Survey of Student Engagement
NSSE Provided Reports
UCF NSSE Engagement Indicator Results
Foundations of Excellence
Topical Briefs
Survey Administration
Staff Functions
OEAS Knowledgebase
Home / Results by Survey - National Survey of Student Engagement - UCF NSSE Engagement Indicator Results
.NET Domain
UCF NSSE Engagement Indicator Results
SAS/IntrNet Domain
Submit

National Survey of Student Engagement

Engagement Indicator Results

The National Survey of Student Engagement (NSSE) is administered periodically at hundreds of four-year colleges and universities to first-year students and seniors who are likely to graduate within 1-2 terms. The NSSE instrument assesses student participation in activities and programs that promote their learning and personal development. At UCF, NSSE is administered every three years in the spring term.

You can explore survey results by selecting the NSSE administration year, student level and the level of analysis. Depending on your selection to the first three drop-downs you may be prompted to make further choices such a college, program or a particular student group. Please note that some subsequent drop-downs will only be populated depending on your selection to the initial drop-downs. For more information regarding the "Level of Analysis," please click on the ? icon.

NSSE year: Please select

Class standing: Please select

Level of analysis: Please select

Figure 1. User view of OEAS Knowledgebase

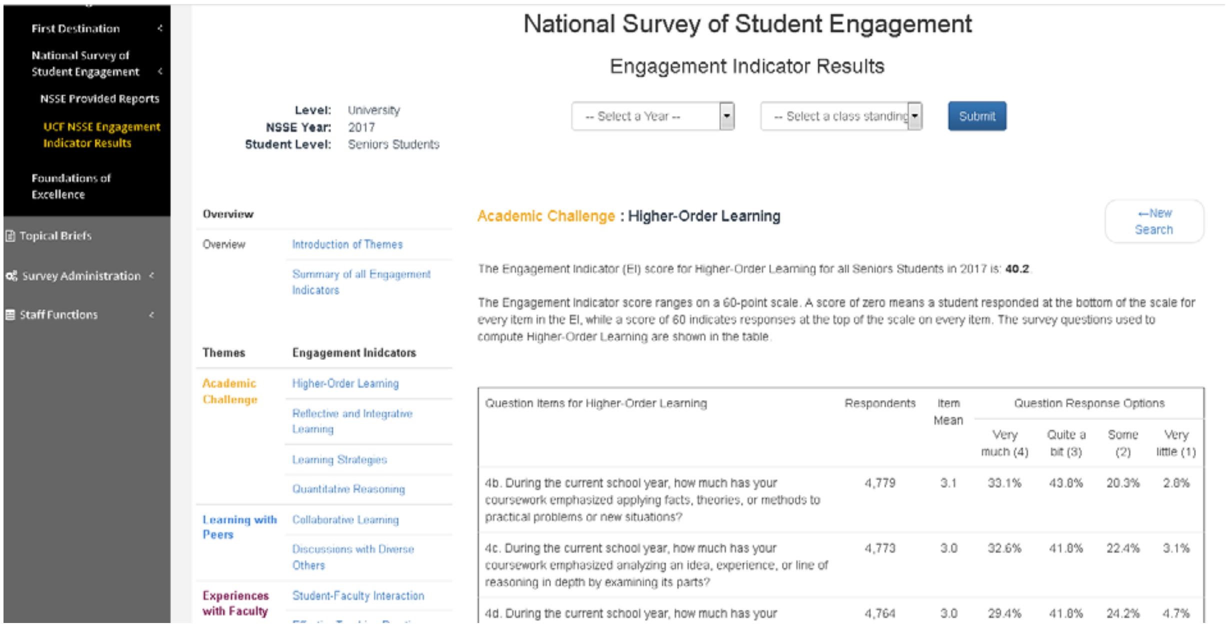
Table 1 shows select features of the web application and the technologies integrated to create them. The following sections in this paper will provide details about how each feature was developed.

Features	Base SAS / SAS IntrNET / SAS Macros ®	.NET	JavaScript	JQuery
Creating uniform layout and page sizing [cross-domain integration]	✓	✓	✓	✓
Dynamically generated cascading dropdown menus	✓		✓	
Printing capabilities from the web application [cross-domain integration]	✓	✓	✓	✓
Tracking web application user behavior [using Google Analytics]	✓	✓	✓	

Table 1. Select features of the web application

AN EXAMPLE OF A DYNAMICALLY GENERATED REPORT IN OEAS KNOWLEDGEBASE FOR NSSE

After an authenticated user logs into the OEAS Knowledgebase system and navigates to NSSE, the user is prompted to select a year of administration, the class standing (First-year or Senior students), the level of analysis (one-way tables or two-way tables) as well as any filters of interest (college or program of enrollment, student demographics). The screenshot in Figure 2 provides an example of how the final report looks in OEAS Knowledgebase. The data are queried and the tables are structured using SAS programs that reside on the SAS/IntrNet server domain. These programs include data steps, PROC SQL, SAS formats, SAS labels, SAS macros, PROC TABULATE and ODS HTML to generate the results.


First Destination
National Survey of Student Engagement
NSSE Provided Reports
UCF NSSE Engagement Indicator Results
Foundations of Excellence
Topical Briefs
Survey Administration
Staff Functions
-- Select a Year --
-- Select a class standing
Submit
←New Search

National Survey of Student Engagement

Engagement Indicator Results

Level: University
NSSE Year: 2017
Student Level: Seniors Students

Overview	
Overview	Introduction of Themes
	Summary of all Engagement Indicators
Themes	**Engagement Inidcators**
Academic Challenge	Higher-Order Learning
	Reflective and Integrative Learning
	Learning Strategies
	Quantitative Reasoning
Learning with Peers	Collaborative Learning
	Discussions with Diverse Others
Experiences with Faculty	Student-Faculty Interaction

Academic Challenge : **Higher-Order Learning**

The Engagement Indicator (EI) score for Higher-Order Learning for all Seniors Students in 2017 is: **40.2**.

The Engagement Indicator score ranges on a 60-point scale. A score of zero means a student responded at the bottom of the scale for every item in the EI, while a score of 60 indicates responses at the top of the scale on every item. The survey questions used to compute Higher-Order Learning are shown in the table.

Question Items for Higher-Order Learning	Respondents	Item Mean	Question Response Options			
			Very much (4)	Quite a bit (3)	Some (2)	Very little (1)
4b. During the current school year, how much has your coursework emphasized applying facts, theories, or methods to practical problems or new situations?	4,779	3.1	33.1%	43.8%	20.3%	2.8%
4c. During the current school year, how much has your coursework emphasized analyzing an idea, experience, or line of reasoning in depth by examining its parts?	4,773	3.0	32.6%	41.8%	22.4%	3.1%
4d. During the current school year, how much has your	4,764	3.0	29.4%	41.8%	24.2%	4.7%

Figure 2. A view of NSSE survey results in OEAS Knowledgebase after user query

INTEGRATING NON-SAS LANGUAGES INTO A SAS PROGRAM AND PRODUCING OUTPUT ON A SAS/INTRNET SERVER

This section will cover some of the basics needed to integrate SAS and other non-SAS languages. Please note that this is not an exhaustive set of steps needed and additional resources can be found at the end of this paper. Nevertheless, there are two key strategies to getting a SAS program to render the query results desired through the web:

1. **Enclose non-SAS languages (HTML, CSS, JavaScript) with put statements** – this will interpret any non-SAS syntax as character strings and produce the desired markup and styles for the output web pages; This strategy is reserved for page layout and function.

```
%macro level_1_text;
data _null_;
file _webout;

put '<div class="col-xs-12">';
put "<p>The Engagement Indicator (EI) score for &categoryname for all
&irclassname in &yearname is: ";
put "<strong>&EI_mean</strong>.<br /><br />";

put 'The Engagement Indicator score ranges on a 60-point scale. A score
of zero means a student responded at the bottom of the scale for every
item in the EI, while a score of 60 indicates responses at the top of
the scale on every item. The survey questions used to compute '
"&categoryname" ' are shown in the table.</p><br / ></div></div>';
run;
%mend level_1_text;
```

2. **Use the Output Delivery System (ODS) –** this is necessary to output any SAS procedure results to the web page; In this case, we have used ODS HTML. Note that the text written above the table in the results is called by the use of a SAS macro called ***%level_1_text***.

```
%level_1_text;
ODS HTML3 body=_webout (no_top_matter no_bottom_matter)
path=&_tmpcat (url=&_replay) rs=none
stylesheet=(URL="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bo
otstrap.min.css");

    proc tabulate data=temp1 style = [just=c vjust=m ];
    class value/ order=internal descending preloadfmt ;
    class qname/order=formatted;
    classlev qname;
    var finished value_dup;
    keylabel all='UCF';

    table qname=''
    ,finished='Respondents'*n=''*f=comma10.  value_dup='Item
    Mean'*mean=''*f=3.1 value='Question Response
    Options'*(rowpctn=''*f=pct.) /
    box={label="Question Items for &categoryname" s=[just=L]} printmiss
    misstext='-';
    weight weight1;
    run;

ODS html3 close;
ODS listing;
```

CREATING UNIFORM LAYOUT AND PAGE SIZING [CROSS-DOMAIN INTEGRATION]

In order to provide a uniform look and feel of the OEAS Knowledgebase site, iframes are used to call SAS pages within the .NET template. The height and width of pages called within iframes can vary. For the width attribute, we've applied the value of 100% as the template width itself adjusts to a user's screen resolution using jQuery and Bootstrap (https://getbootstrap.com/docs/4.3/getting-started/download/) JavaScript libraries. The height attribute can vary widely and unfortunately the jQuery and Bootstrap JavaScript libraries don't have this functionality built in so the iframe Resizer JavaScript libraries were added.

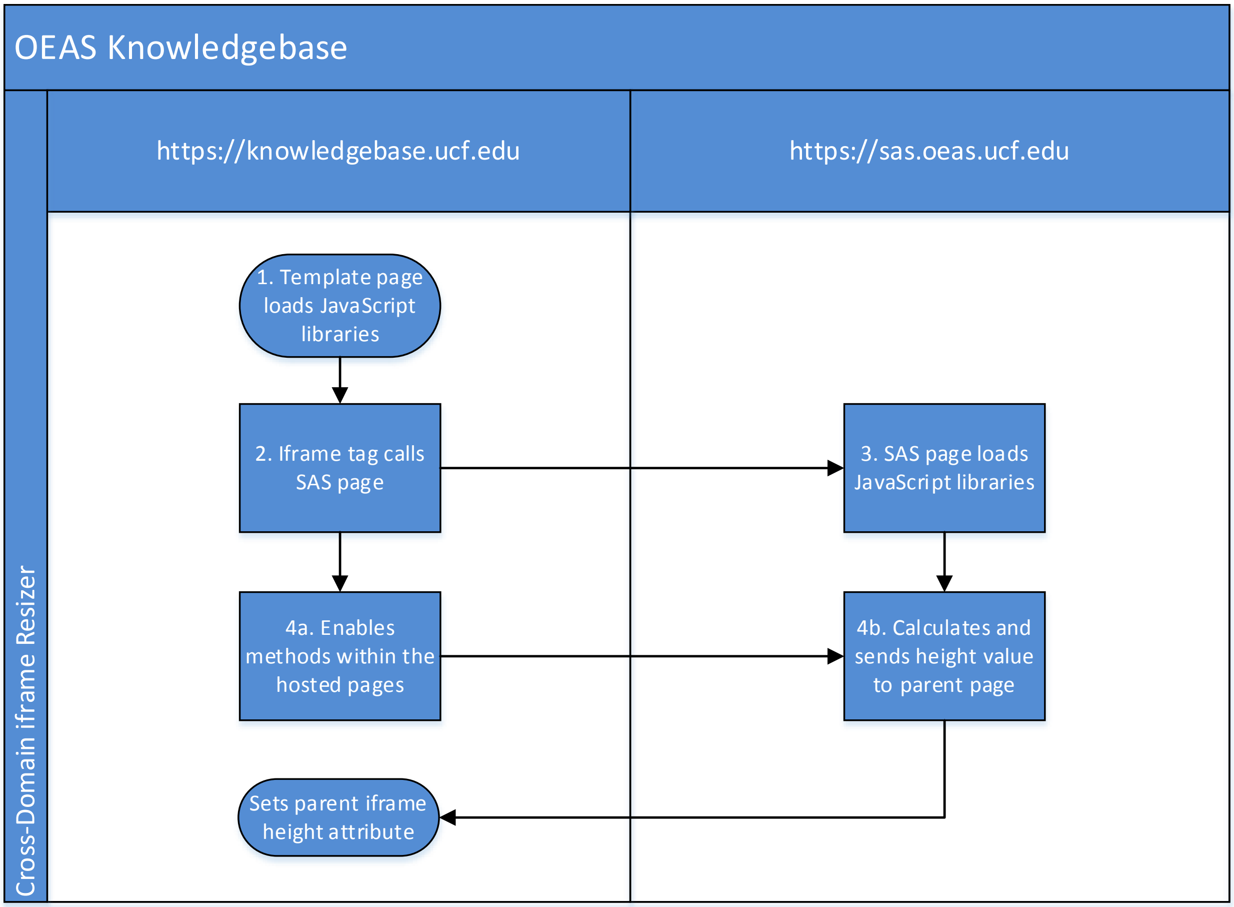

OEAS Knowledgebase
Cross-Domain iframe Resizer
https://knowledgebase.ucf.edu
https://sas.oeas.ucf.edu
1. Template page loads JavaScript libraries
2. Iframe tag calls SAS page
3. SAS page loads JavaScript libraries
4a. Enables methods within the hosted pages
4b. Calculates and sends height value to parent page
Sets parent iframe height attribute

Figure 3. Cross-Domain iframe Resizer Work Flow Diagram

1. Both the .NET template and SAS pages require a script reference to the jQuery library before the iframe Resizer libraries. The following script reference is located in the html head tag on both the .NET template and SAS web page. Please recall that non-SAS syntax has to be enclosed within "put" statements as discussed in prior section.

   ```
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
   </script>
   ```

   The iframe Resizer libraries can be downloaded from the link below.

   https://github.com/davidjbradshaw/iframe-resizer

In the .NET, the iframe Resizer library is referenced in the html tag.

```
<script src="../js/plugins/iframeResizer/iframeResizer.min.js"></script>
```

2. The .NET template iframe tag calls the SAS program.

```
<iframe src="<%= getSASServer%>/survey/NSSE/HTMLPages/nsse_input.htm?z=<%=
getUserID()%>" width="100%" scrolling="no" frameborder="0" name="NSSEUCFEIContent"
id="NSSEUCFEIContent"></iframe>
```

3. The SAS pages reference the iframe Resizer content window library as such.

```
<script type="text/javascript"
src="XXXXXXXXXXXX/iframeResizer.contentWindow.min.js"></script>
```

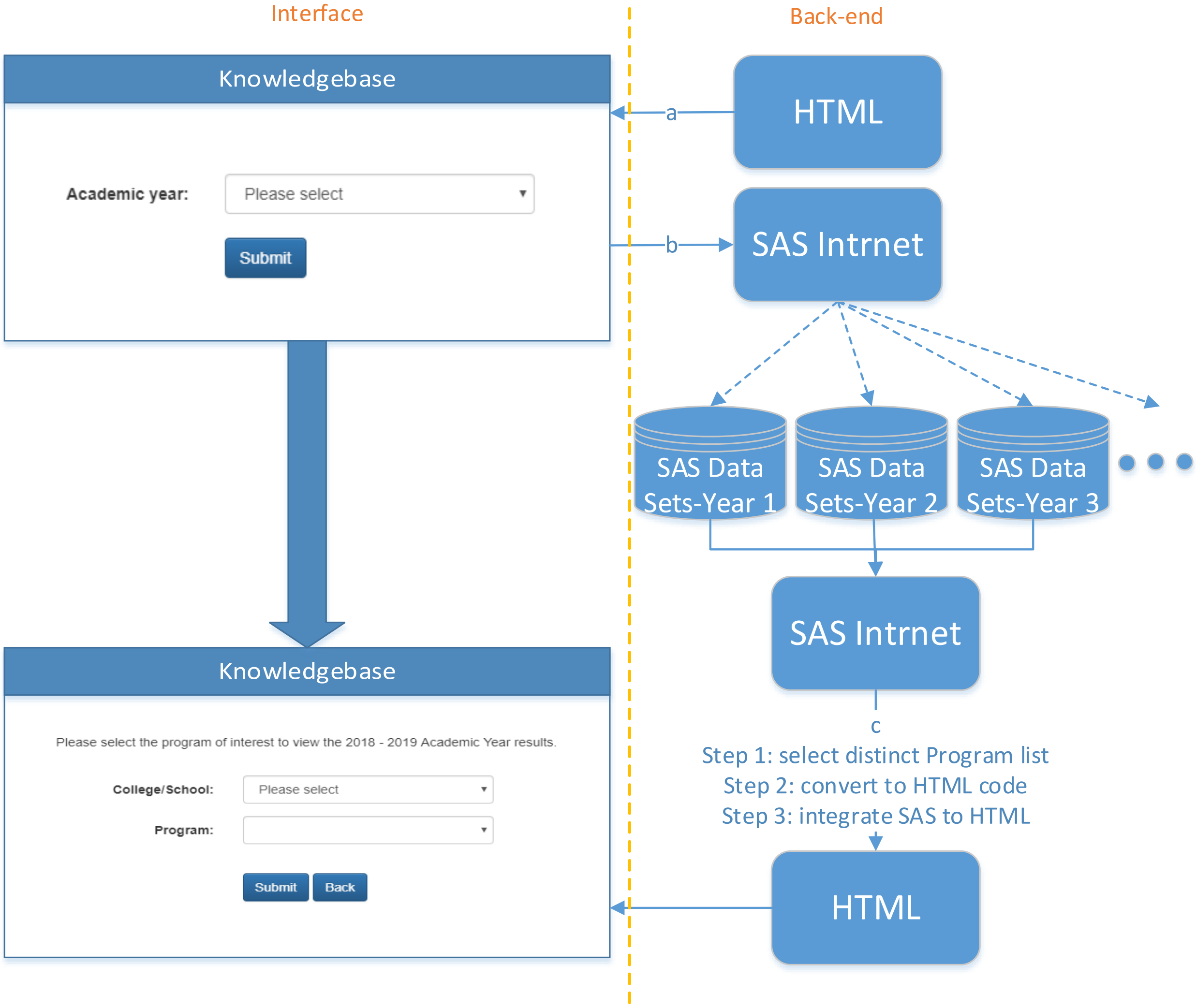
4. In addition to the script libraries referenced in the .NET template head tag, the following function is added after the iframe and before the HTML end tag that enables the methods within the iframe SAS pages being hosted.

```
<script type="text/javascript">
       iFrameResize({
       log: true,                     // Enable console logging
       enablePublicMethods: true,     // Enable methods within iframe hosted page
       resizedCallback: function (messageData) { // Callback fn when resize is
       received
              },
       messageCallback: function (messageData) { // Callback fn when message is
       received
              },
       closedCallback: function (id) { // Callback fn when iFrame is closed
              }
       });
</script>
```

DYNAMICALLY GENERATED CASCADING DROPDOWN MENUS

Cascading dropdown lists are common in web interfaces to allow users to select specific parameters for their queries. The second dropdown list will depend on what the user selects from the first dropdown list. For example, the use of cascading dropdowns are employed in this web application so that users can view results for students enrolled in an academic major within a particular college.

A problem may ensue if there is a long subcategory list which needs maintenance and recurring edits. Academic majors can change at an institution over time for various reasons such as re-organizing, activating or deactivating certain majors, etc. Furthermore, each survey administration results in different student respondents and not all majors are represented in the sample every year. This results in new majors being added or deleted to the dropdowns with each new administration year. These changes are hard to maintain and need a lot of effort to keep accurate. To solve this problem, the information needed to populate the dropdowns can be pulled directly from the data set rather than listed manually.



Please select the program of interest to view the 2018 - 2019 Academic Year results.

Step 1: select distinct Program list
Step 2: convert to HTML code
Step 3: integrate SAS to HTML

Figure 4. Screenshot of the dynamic cascading dropdowns in the in OEAS Knowledgebase

a. **Passing user input to the SAS program**: The initial page is written as an HTML page (Fig. 4-a) which allows a user to select a NSSE year of interest. The selected parameter, “**year”**, is sent to the underlying SAS program “**programselection.sas**” and is used to identify the SAS library and SAS data sets to be used for the query (Fig. 4-b). These values are highlighted in the HTML code below. As mentioned previously, these parameters are passed within the uniform resource locator (URL) query string.

```
<form class="form-horizontal" name="form1" method="get"
action="https://sasqa.oeas.ucf.edu/scripts/broker.exe"
onsubmit="return Form1_Validator(this)">
<input type="hidden" name="_debug" value="0" />
<input type="hidden" name="_service" value="survey" />
<input type="hidden" name="_program" value="programselection.sas"/>
<input name="z" value="0" type="hidden" />
<select class="form-control" name="year" id="term">
<option value="0">Please select</option>
<option value="2017">2017</option>
<option value="2014">2014</option>
                    .
                    .
</select>
```

b. Since different years of NSSE administration data are housed in different directories, the “year” a user selects will determine the libname statement in the SAS program

```
%if &year = 2014 %then %do;
  libname NSSE “<directory for SAS library in 2014>”;
%end;
%else %if &year = 2017 %then %do;
  libname NSSE “<directory for SAS library in 2017>”;
%end;
```

c. The SAS program “**programselection.sas**” queries a list of the colleges and academic programs for all NSSE respondents and places the list into a SAS macro to be reference later in an HTML dropdown list

Step 1: SAS – PROC SQL queries program list from that year’s NSSE table

```
PROC SQL;
create table prolist as
select distinct(ACAD_PLAN_DESCR), ACAD_PLAN,
ACADGROUP_DESCR, ACADGROUP
from NSSE;
QUIT;
```

Step 2: Creates a character variable that captures the HTML syntax needed in a dropdown menu and then save them all listed as one SAS macro variable. In this example the DATA STEP was used with a CAT function, although this could have also been implemented in the PROC SQL statement in Step 1. The CAT function does not remove the leading and trailing blanks, so a TRIM function was also used. Final values of the character strings in “**drop2**” would look something like this for the Art BA program in the College of Arts and Humanities (CAH) - “<option class='CAH' value=' ARTBA'> Art (BA) </option>".

```
DATA prolist;
SET  prolist;
length drop2 $200.;
```

```
drop2 = CAT ("<option class='",TRIM(ACADGROUP),"'
value='",TRIM(ACAD_PLAN),"'>",TRIM(ACAD_PLAN_DESCR),"</option>");
RUN;

PROC SQL;
SELECT drop2 INTO: droplist_pro SEPARATED BY ' ' FROM prolist
order by ACAD_PLAN_DESCR;
QUIT;
```

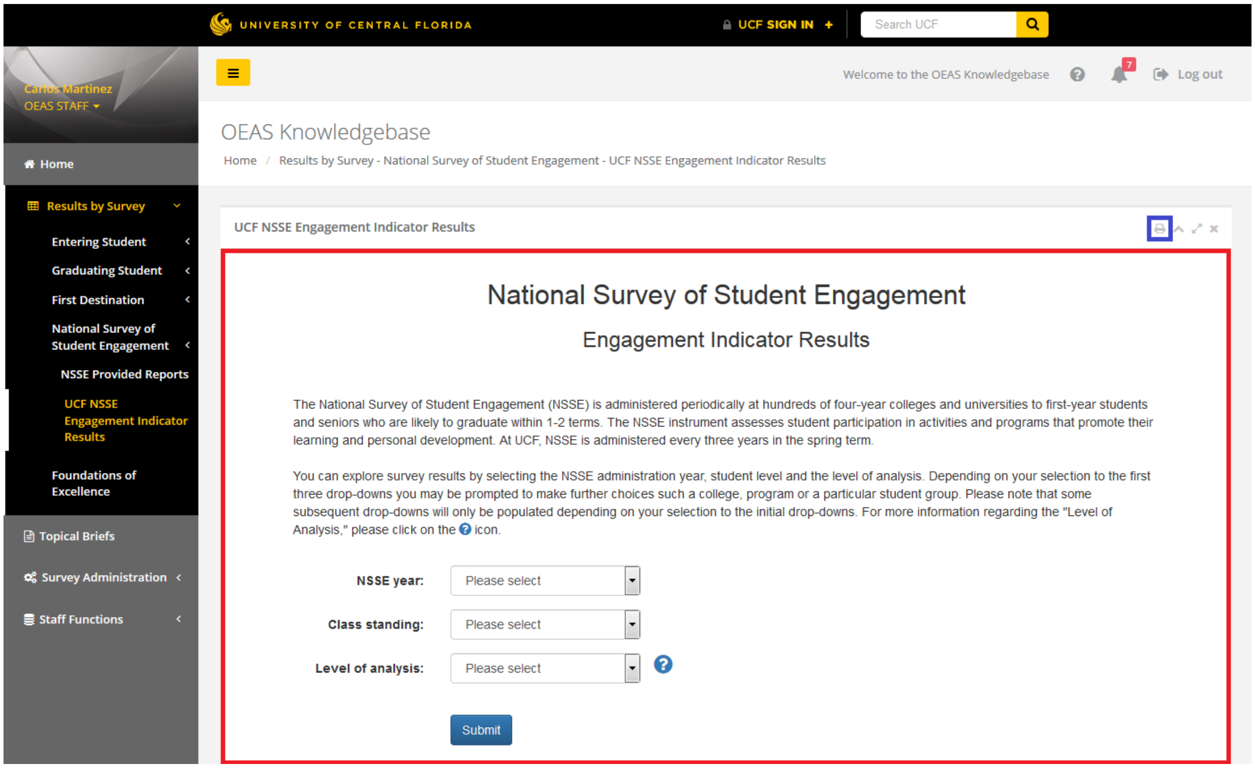
Step 3: HTML – integrate SAS macro variable "**droplist_pro**" into HTML.

```
 put '<select class="form-control" name="items1" id="items1">
 <option class="CAH" value="0">Please select</option>
 <option class="CBA" value="0">Please select</option>
                .
                .
 <option class="UGST" value="0">Please select</option>';
put "&droplist_pro";
put '</select>';
```

PRINTING CAPABILITIES FROM THE WEB APPLICATION [CROSS-DOMAIN INTEGRATION]

In some cases the passing of data using a URL query string is not enough, like the ability to print content of the SAS domain. For these specific cases, a JavaScript library passes data through a proxy between two domains.

The screenshot below is of the OEAS Knowledgebase. The section in the red rectangle contains SAS/IntrNet web pages from https://sas.oeas.ucf.edu that are called from an iframe tag located on an https://knowledgebase.ucf.edu .NET template page. The icon with the blue square around it is located on https://knowledgebase.ucf.edu. In order for a user to print the iframe object from the printer icon, a combination of the porthole and jQuery JavaScript libraries are used.


UNIVERSITY OF CENTRAL FLORIDA
UCF SIGN IN +
Search UCF
Carlos Martinez
OEAS STAFF
Welcome to the OEAS Knowledgebase
7
Log out
OEAS Knowledgebase
Home / Results by Survey - National Survey of Student Engagement - UCF NSSE Engagement Indicator Results
Home
Results by Survey
Entering Student
Graduating Student
First Destination
National Survey of Student Engagement
NSSE Provided Reports
UCF NSSE Engagement Indicator Results
Foundations of Excellence
Topical Briefs
Survey Administration
Staff Functions
UCF NSSE Engagement Indicator Results

National Survey of Student Engagement

Engagement Indicator Results

The National Survey of Student Engagement (NSSE) is administered periodically at hundreds of four-year colleges and universities to first-year students and seniors who are likely to graduate within 1-2 terms. The NSSE instrument assesses student participation in activities and programs that promote their learning and personal development. At UCF, NSSE is administered every three years in the spring term.

You can explore survey results by selecting the NSSE administration year, student level and the level of analysis. Depending on your selection to the first three drop-downs you may be prompted to make further choices such a college, program or a particular student group. Please note that some subsequent drop-downs will only be populated depending on your selection to the initial drop-downs. For more information regarding the "Level of Analysis," please click on the ? icon.

NSSE year: Please select

Class standing: Please select

Level of analysis: Please select

Figure 5. Screenshot of the OEAS Knowledgebase print feature

The following flowchart describes the cross-domain communication between OEAS Knowledgebase and SAS/IntrNet web pages.

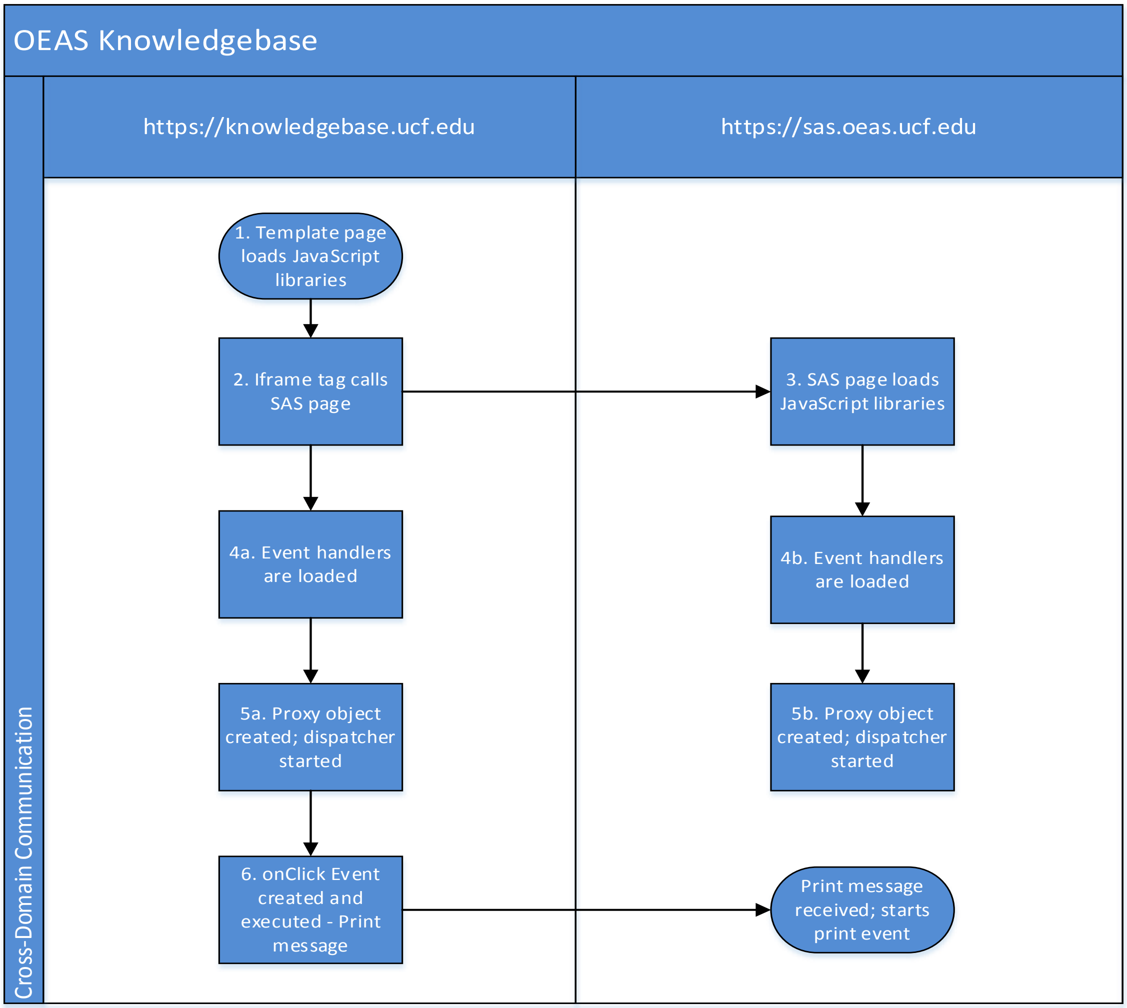

OEAS Knowledgebase
Cross-Domain Communication
https://knowledgebase.ucf.edu
https://sas.oeas.ucf.edu
1. Template page loads JavaScript libraries
2. Iframe tag calls SAS page
3. SAS page loads JavaScript libraries
4a. Event handlers are loaded
4b. Event handlers are loaded
5a. Proxy object created; dispatcher started
5b. Proxy object created; dispatcher started
6. onClick Event created and executed - Print message
Print message received; starts print event

Figure 6. Cross-Domain Work Flow Diagram

1. Both the .NET template and web page on the SAS/IntrNet domain require a script reference in the html head tag to the jQuery library before the porthole library. Please recall that non-SAS syntax has to be enclosed within “put” statements as discussed in prior section.

   ```
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
   </script>
   ```

   The porthole library can be downloaded from the link below.

   https://github.com/ternarylabs/porthole/

   In the .NET template, the porthole library is referenced in the HTML tag.

   ```
   <script src="../js/plugins/porthole/porthole.min.js"></script>
   ```

2. The .NET template iframe tag calls the web page on the SAS/IntrNet domain.

   ```
   <iframe src="<%= getSASServer%>/survey/NSSE/HTMLPages/nsse_input.htm?z=<%=
   getUserID()%>" width="100%" scrolling="no" frameborder="0" name="NSSEUCFEIContent"
   id="NSSEUCFEIContent"></iframe>
   ```

3. The SAS pages reference the porthole library as such.

   ```
   <script src="https://sas.oeas.ucf.edu/survey/KB_HTML/SHARED/porthole.min.js">
   </script>
   ```

4. In addition to the script libraries referenced in the .NET template, the following functions are created to handle the printing of the iframe object. Since we are using a .NET template page, the getSASServer is a method call to retrieve the URL of the SAS server domain.

   ```
   a.  <script type="text/javascript">
          var guestDomain = '<%= getSASServer%>';

          function onMessage1(messageEvent) {
              if (messageEvent.origin == guestDomain) {
                  if (messageEvent.data["print"]) {
                  }
              }
          }

       </script>
   b.  <script type="text/javascript">
       //execute when the page is ready
              var windowProxy;
              function onMessage(messageEvent) {
                   if (messageEvent.data["print"]) {
                       window.print();
                   }
              }
       </script>
   ```

5. Right before the closing HTML tag on the .NET template, the following function is added to create a proxy window to send and receive data from the iframe.

   ```
   a. <script type="text/javascript">
          var windowProxy1;

          function nsseucfeiPage() {
                 // Create a proxy window to send to and receive message from the
                 guest iframe
                 windowProxy1 = new Porthole.WindowProxy(guestDomain +
                 '/survey/KB_HTML/SHARED/proxy.html', 'NSSEUCFEIContent');
                 windowProxy1.addEventListener(onMessage1);
           }

           $(document).ready(nsseucfeiPage);


          Sys.WebForms.PageRequestManager.getInstance().add_endRequest(nsseucfeiPage)
          ;
       </script>
   ```

   b. The SAS pages contain the following script in the html head tag which enables the proxy listener that is called from the .NET template.

   ```
   <script type="text/javascript">
       window.onload = applyProxy;
   ```

```
   //Adds proxy listener
   function applyProxy() {

           windowProxy = new
           Porthole.WindowProxy("https://knowledgebase.ucf.edu/js/plugins/port
           hole/proxy.html");
           windowProxy.addEventListener(onMessage);
    }
</script>
```

6. The print icon in the .NET template has an anchor that calls the proxy window.

```
<a onclick="windowProxy1.post({ 'print': 'me' })" >
   <i class="fa fa-print" title="Print"></i></a>
```

TRACKING WEB APPLICATION USER BEHAVIOR [USING GOOGLE ANALYTICS]

To improve the functionalities of the web application and provide meaningful information to the end user it is important to have an insight into the user behavior when interacting with the application. Google Analytics captures URL query strings, among a wealth of other data, and stores them so they can be analyzed. The data captured includes a User ID value, the survey of interest, other parameters from the query being run. All these values are passed in the URL query string. This User ID value is generated in a Microsoft SQL database when a user is created in OEAS Knowledgebase. When a user is added, it is authenticated against an internal Microsoft Domain using Lightweight Directory Access Portal (LDAP). Upon logging into the OEAS Knowledgebase site, the User ID is queried and passed through to the SAS web pages as the parameter “z”.

1. The .NET template iframe tag calls the SAS web page. Since we are using a .NET template page, the getUserID is a method call to retrieve the UserID stored in the OEAS Knowledgebase database.

```
<iframe src="<%= getSASServer%>/survey/NSSE/HTMLPages/nsse_input.htm?z=<%=
getUserID()%>" width="100%" scrolling="no" frameborder="0"
name="NSSEUCFEIContent" id="NSSEUCFEIContent"></iframe>
```

2. The value of the z parameter that is passed to the surveyname_input.htm pages on the SAS domain is collected as a hidden input tag also named “z”.

```
<input name="z" value="0" type="hidden" />
```

The value of the input tag is assigned by a script that’s been added to the end of the *surveyname*_input.htm pages.

```
<script>
    $(document).ready(function () {
        $('input[name="z"]').val(getUrlParameter("z"));
    });
</script>
```

3. In order to utilize Google Analytics, a free account must be created (https://analytics.google.com/analytics/web/). Once the account is created, the domain that is to be tracked must be submitted to their system. A snippet of JavaScript code is then provided by Google that must be placed on every page, template, or in our case, the footer macro that is called each time a SAS web page is rendered.

```
/* Footer for the bottom of the NSSE Survey web app;*/

%macro s_footer;
data _null_;
   file _webout;
   %if &Prod = 1 %then %do;
   /********JavaScript code provided by Google Analytics****/
   put '<script>

   (function(i,s,o,g,r,a,m){i["GoogleAnalyticsObject"]=r;i[r]=i[r]||fun
   ction(){
        (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
   Date();a=s.createElement(o),

   m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insert
   Before(a,m)
        })(window,document,"script","https://www.google-
   analytics.com/analytics.js","ga");

  ga("create", "XX-########-#", "auto");
  ga("send", "pageview");

      </script>';
  %end;
  put '</body>';
  put '</html>';
run;

%mend;
```

4. When the user submits their input in the *surveyname*_input.htm pages, the SAS Internet broker service is called, first using a JavaScript validation function, passing the “z” value in the URL query string. The SAS program and service are listed as hidden tag values in the *surveyname*_input.htm pages and also passed in the URL query string.

```
<form class="form-horizontal" name="form1" method="get"
action="xxxxxxx/scripts/broker.exe" onsubmit="return Form1_Validator(this)">
<input name="_service" value="survey" type="hidden" />
<input name="_program" value="nsse.nsse_index.sas" type="hidden" />
```

5. The SAS *surveyname*_index.sas files instantiate a global variable that is referenced in all anchor tags.

```
%global z;
put '<div class="col-xs-2"> <ul class="pager">';
put ' <li class="next">';
put '<a href="';
put "&address_html"; /*defined in server_references.sas*/
put '/NSSE_input.htm?z=';
put "&z";
put '#top">';
put'<span aria-hidden="true">&larr;</span>New Search</a></li>';
put ' </ul></div>';
```

CONCLUSION

Integrating technologies to develop a fluid dynamic web application can sometimes be a difficult task considering possible roadblocks such as developer background, expertise, and knowledge of programming languages. SAS/IntrNet ® platform, JavaScript, HTML, and .NET have made it possible to deliver a seamless and dynamic web application consisting of cascading input query pages, custom reporting and easy-to-navigate reporting interface.

RECOMMENDED READING

Cano, Gabe. 2008. “Getting Started with SAS/IntrNet® - A Quick and Dirty Tutorial.“ Proceedings of the 2008 SAS Global Forum. San Antonio, Texas.

Palbicki, Jeremy. 2016. “Rapid Web Services using SAS/IntrNet® Software, jQuery AJAX, and PROC JSON.” Proceedings of the 2016 SAS Global Forum. Las Vegas, Nevada.

Redelings, Matthew. 2010. “Query Creation Wizards Using SAS® IntrNet” Proceedings of the Western Users of SAS Software. San Diego, California.

CONTACT INFORMATION

Carlos Martinez
University of Central Florida
carlos.martinez@ucf.edu

Uday Nair
University of Central Florida
uday.nair@ucf.edu

Wen Jiang
University of Central Florida
wen.jiang@ucf.edu