

Building a SAS Intranet Site

Tim Williams, PRA International, Charlottesville, VA

ABSTRACT

Intranets are now common in both large and small organizations. These internal websites provide a way to present vast quantities of information in an organized and searchable form. This paper describes the construction and implementation of an intranet site developed by the Analysis Programming staff of PRA International in Charlottesville, VA. Methods for content development and display of SAS coding examples are illustrated. Management of SAS coding projects is discussed in terms of how a database-driven website can increase the efficiency of a programming team and lead to a better deliverable product. It is hoped that this paper will stimulate discussion on how intranets may better serve SAS programmers in the pharmaceutical and other industries.

INTRODUCTION

When I recently joined PRA International, a well-established Contract Research Organization (CRO), I found I had a great deal to learn about the industry. While I was an experienced SAS user, this was my first exposure to the world of pharmaceutical research in the United States. I needed to learn new standard operating procedures and coding conventions. Experienced personnel were often required to assist me in locating information and explaining policies. The situation called out for a well-organized, indexed and searchable resource for staff.

At the time of my employment, PRA was embarking on the implementation their of intranet. Initial development of static HTML pages had begun (Phase 1) with a planned move to dynamically generated content during Phase 2. These early developmental stages provided an excellent opportunity for departments to define their needs, conceptualize site content, and contribute to decisions regarding choices of technology.

This paper describes the initial design of a static HTML intranet site for the Analysis Programming Department at PRA. It also includes a discussion on how processes will be streamlined when a dynamic, database-driven site is implemented. The paper will not discuss issues such as website style, navigation, scripting, or comparisons between technologies and approaches. These issues are beyond the scope of the paper and are often dependent upon company policy and individual preferences.

DEVELOPMENT STRATEGY

Winning the support of management and staff was critical to the project. The staff already had heavy workloads, so it was crucial to build a strong case for the departure from client-oriented project work. Managers and staff met early in the project to determine what processes and procedures could be targeted for rapid content development and benefit most from web integration.

Our staffing and workload constraints lead us to adopt the staged delivery model of software development (McConnell, 1996). Staged delivery, also known as incremental implementation,

makes content available to users in successive stages as the site is developed. In this way our staff and management saw tangible benefits of the intranet without having to wait for its full implementation. The first content was purely informational; it told the company who the Analysis Programming Department is and what they do. Next, we quickly realized our second goal of providing a resource for department staff. Early successes helped to strengthen support for later, more ambitious aspects of development.

Our overall objective was to start with static content while structuring the site to facilitate the switchover to database-driven dynamic pages (using ASP and Oracle) in the near future. Several staff were already familiar with basic HTML or were able to learn it in a short period of time. The anticipated delay in moving to dynamic content will allow staff to obtain the skills and training they need.

DEVELOPING CONTENT

The PRA Webmaster developed HTML templates, style sheets, and initial site structure for all departments. These templates provided an excellent launch pad for rapid development because they freed individual developers from the time and effort required when starting from scratch. Individual departments selected development teams to identify content requirements and start site construction.

We organized available materials into a series of easy-to-navigate HTML pages to provide immediate benefits to staff. Many of the vital components already existed in electronic form and were easily linked to HTML pages. Such items included:

- Standard operating procedures
- FDA Electronic Submission Guidelines
- Coding conventions
- Project specific information

HTML content was searchable directly after publishing to the web server because we used Microsoft FrontPage 2000 and published to a Microsoft Internet Information Server 4.0. (IIS 4.0) with the FrontPage Server Extensions installed. Additional benefits over the pre-existing documents were gained by including hyperlinks to related site content. For example, links to a Table of Acronyms has proven to be a useful tool for employees new to the industry.

KEY CONTENT

Essential content was identified by meeting with interested department staff and storyboarding the website. We defined the following areas for early development:

- Training in company-specific SAS Coding Methods
- Standard Operating Procedures
- Table of Acronyms
- SAS Content
 - SAS help documentation
 - External SAS resources

- Search facility for SAS discussions (see below)
- SAS functions reference
- Standardized "in-house" SAS macros and code modules
- Code examples contributed by staff

SEARCHING THE SAS-L ARCHIVES AND COMP.SOFT-SYS.SAS

One of the most popular items on our website is the ability to conveniently search both the UGA SAS-L Listserv and **comp.soft-sys.sas** through the website through the Google search facility at <http://groups.google.com>. Google Inc. acquired the Deja.com Usenet Discussion Service in February 2001. The service provides a means to search millions of Usenet messages dating back to 1995.

Instead of providing simple links to the internet home pages of these searchable archives, we placed the search forms within our intranet, thus removing the additional step of loading the search page from the remote server. Our staff simply enters the search text along with other optional parameters and the form is then submitted to the remote search engine. The browser returns the search results page when processing is complete.

The code below illustrates how simple it can be to incorporate such functionality into a web page. In this example, **comp.soft-sys.sas** is searched for articles containing a list of all the words entered on a form:

```
<form method=GET
  action= "http://groups.google.com/groups"
  name=f>

  <h3>Search comp.soft-sys.sas</h3>
  for all of these words:
  <input type=text value="" name=as_q
    size=25>
  <br>
  <input size=30 type=hidden
    value="comp.soft-sys.sas" name=as_ugroup>
  <input type=submit value="Google Search">

</form>
```

At the time of this writing, groups.google.com supports the following fields on their "Advanced Groups Search" form:

- All words in a list (as in the example above)
- Any of the words in a list
- An exact phrase
- Subject
- Author
- Message ID
- Language

Further options are also available to limit the search to a range of dates, return only a specific number of items, and sort the results by either relevance or date.

Developers can easily add these fields to their own form by locating the field names on the "Advanced Groups Search" page at <http://groups.google.com>. Simply use the View | Source menu on Internet Explorer when the remote page is loaded. A similar method was used to implement a form to search the UGA SAS-L Listserv from our intranet.

PRESENTING SAS CODE ON THE INTRANET

Formatting of the original code lines should be preserved when presenting SAS code examples on the web. This allows users to cut-and-paste code from the web into a SAS code editor. Similarly, contributors to the website need to cut-and-paste SAS code into FrontPage (or another HTML editor) while preserving formatting. Users of the website should be able to quickly identify SAS code through the use of distinct fonts and colors. All of these elements are available through the use of HTML tags and style sheets.

<PRE> TAG

The HTML tag <PRE> defines a section of text that the browser will render in exactly the same character and line spacing as it appears in the HTML source document. It is this tag that allows the programmer to cut-and-paste SAS code from the SAS editor (or other text editor) directly into HTML while preserving spacing and indentation of the code lines. Some authors use this tag in conjunction with the <BLOCKQUOTE> tag that helps to delimit the code section by applying special indentation and formatting. Our approach was to instead use our own user-defined tag <CODE> to encapsulate SAS code blocks and render formatting through application of styles to that tag.

STYLE SHEETS AND THE <CODE> TAG

Cascading Style Sheets (CSS) provide a powerful and easy way to provide a consistent font, font size, color and other properties for HTML pages, sections of pages, or an entire website. The web author can not only control the presentation attributes of standard HTML tags but can also implement user-defined tags for their own use. Styles can be applied to HTML tags using inline styles, document-level styles, and external style sheets. Use of external style sheets is often preferred because it allows the web author to apply global style changes to a website by editing only one file. Styles "cascade" down from the external sheets through local document styles down to inline styles.

We developed a style sheet for SAS program code to enhance the style sheet supplied by the PRA Webmaster. We chose to distinguish the program logs, outputs, comments, and required program parameters from surrounding text on the web page. We applied additional styles to code linked to documentation through either a mouse-over or a mouse-click. Our new style sheet was linked to all documents containing SAS code using the following line in the header section of HTML pages:

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

Styles were then applied to the user-defined <CODE> tag by specifying different values for the CLASS parameter as shown in the table below.

Classes of the user-defined <CODE> tag

CLASS=	Used for display of:
comment	Comments within code, typically code lines with /* */ , * or %* comment styles
change	Program code that should be changed by the user prior to implementation. E.g.: a random number seed.
explain	Text explaining a code segment
footnote	Designates footnotes that are not part of the code but are used to label code for explanation in a footnote

log	SAS log
out	SAS output
prog	SAS program code

The following code illustrates the use of the <PRE> and <CODE> tags :

```
<pre>
<code class="prog">
data random1(drop=i);
  do i = 1 to 10;
    random=ranuni(
      <code class="change">54321</code>);
    output;
  end;
run;
</code>
</pre>
```

In addition to uniquely rendering the types of SAS code being presented, it is also useful to identify sections of code that contain explanations or documentation available through mouse-overs or hyperlinks within the code itself. Styles were therefore defined for use with the HTML <A> tag to enable users to identify these areas on the screen. Typically, these <A> tags will appear enveloped within <CODE> tags for the presentation of SAS code and output.

Classes of the HTML tag <A>

CLASS=	Used for display of:
example	Code is hyperlinked to additional examples illustrating the same concept or program statements.
explain	Code displayed with this parameter is hyperlinked to an explanation or documentation
footnote	Code is hyperlinked to additional information in a footnote.
mouseExplain	An explanation of the code will appear on the page when the mouse pointer is moved over the code section.

The code below illustrates the use of the <A> tag to display the ranuni function as an HTML hyperlink which, when clicked, takes the user to the HTML document sasFunctions.html that contains additional documentation about the function.

```
<pre>
<code class="prog">
%let rannum=<a
href="./sasFunctions.htm#ranuni"
class="explain">ranuni</a>(1234);
</code>
</pre>
```

USING SAS TO GENERATE AND UPDATE HTML PAGES

Many papers have been presented in past conferences describing how to generate HTML output from SAS prior to the introduction of ODS in Version 8 (Bahler 1999, 1998 ; Pope 1997). SAS can also be used to update existing HTML files that are subsequently published to the intranet. One way to

accomplish this is to develop a template page containing HTML comment tags that delimit a section where a SAS program will insert output. No text between the start of the HTML comment tag "<!--" and the end of the tag "-->" is displayed by a browser, so the comment tag provides a convenient and invisible marker for use by a SAS program.

[Top of the HTML document omitted]

```
<h2>SAS Generated Table Follows</h2>
<!-- Start SAS Insertion -->
  <!-- this text will be replaced by SAS -->
<!-- End SAS Insertion -->
<em>Table generated using SAS 6.12</em>
```

[Bottom of the HTML document omitted]

It is then a simple task to develop a SAS program that reads the HTML page one line at a time and writes it back out until the insertion point is found. When the program finds the insertion point a routine is called that inserts SAS output into the HTML document.

Once the SAS program has finished inserting HTML, it will read in (but not write out) lines from the source HTML file until it reads in a line denoting the end of the insertion area:

```
<!-- End SAS Insertion -->
```

Using this approach, any text that exists in the HTML template between the start and end of the insertion point is not written to the HTML output file. When the end of the insertion point is found, the SAS program reads in and writes out each line until the end of the input file is reached.

The completed HTML output file must then be published to the web server before it can be viewed. Although inconvenient, this method has provided a way to extract data from Oracle databases for display on our intranet, using SAS as an intermediate step.

THE CHALLENGES OF MICROSOFT FRONTPAGE

Microsoft has made many improvements over early versions of FrontPage but it can still be problematic for the HTML source code enthusiast. Care must be taken when developing the "template" pages mentioned above; opening them into FrontPage may result in pages with lines longer than the 200 character limit of SAS version 6.12 and earlier. The FrontPage visual interface also has an undesirable habit of inserting many proprietary tags and redundant formatting tags that make editing of the source code difficult.

However, if a developer employs patience and ingenuity, FrontPage can make the presentation of SAS code easier (and thereby encourage other programmers to submit code examples to the website). One such example is the ability to easily apply the user-defined tags for formatting SAS code. Ideally, a developer would select code with a mouse in FrontPage's visual editor, then click on a button to apply the custom style to the selection.

This functionality can be added to FrontPage by defining macros in the Visual Basic Editor and assigning these macros to buttons

on a toolbar. An example of a Visual Basic macro that applies the <CODE CLASS="prog"> tag reads as follows:

```
Sub CODEProg()  
    ' Apply the <CODE CLASS="PROG"> tag  
    ' to multiple selected lines  
    Dim objTxtRange As IHTMLTxtRange  
    Dim myHTML As String  
    'Range of selected text  
    Set objTxtRange =  
    ActiveDocument.selection.createRange  
    myHTML = "<code class=" & ""prog"" & ">"  
    & objTxtRange.htmlText & "</code>"  
    objTxtRange.pasteHTML (myHTML)  
End Sub
```

FUTURE DIRECTIONS

An intranet will truly come into its own when it uses dynamic content. The next step in our development process is to move toward dynamic content generation using Active Server Pages (ASP) and a database back-end.

In our department we have well-defined quality assurance methods for code development, team coordination, and quality control. These three elements currently exist as separate entities that could be joined together using web technology.

Our SAS programs contain a standardized header section that includes information such as :

- Author
- Program abstract
- Date of code completion
- Date of code validation
- Author of code validation
- List of input and output files
- List of macros called
- Notes
- Amendments

Headers serve not only as a guide to our programmers but also to our clients who often request the source code as part of the deliverable product. Much of the program header information is repeated in a Microsoft Excel "Table of Programs" (TOP) spreadsheet. Our programmers use the TOP to coordinate code development while providing additional documentation for each SAS program.

After programming is complete, each program is validated using a rigorous Quality Control Checklist. As a series of MicroSoft Word documents (one for each SAS program) the QC Checklists represent a third element that disconnects and adds redundancy to the process (FIGURE 1).

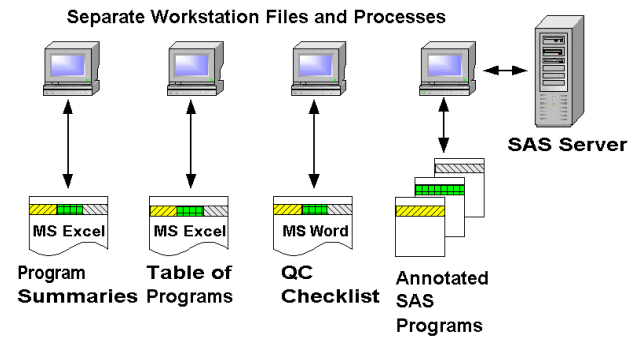


Figure 1 : SAS programs and related files prior to database integration.

Related elements of the coding process can be integrated into a web-based system (FIGURE 2). A centralized database will hold information about each program for display and updating through dynamically generated web pages. These pages will allow developers to enter information about a project and its associated files. QC documents will no longer exist as disconnected elements, but will share fields with the Table of Programs and SAS Headers in a database. Redundancy of data entry will be greatly reduced and the entire process will become streamlined due to the centralization of information. Programmers will be able to spend more hours on client oriented project work and less time laboriously filling out redundant fields during code validation and project management.

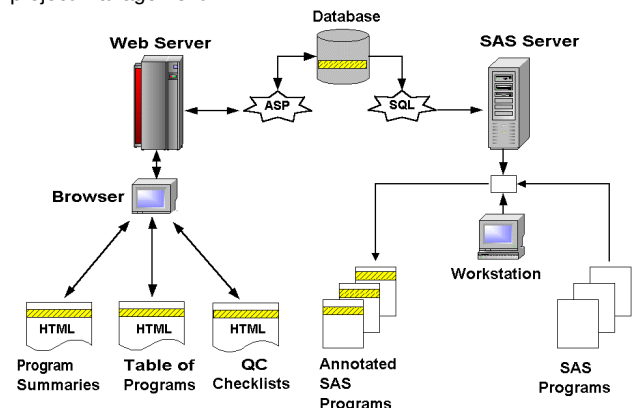


Figure 2 : Integrated database approach to managing SAS programs and related files

The approach to the SAS program headers requires further investigation. One option may be to have a reduced header during code development that contains only a few key fields needed to link it to the database. One could foresee the placement of a hyperlink that, when clicked in the editor, would launch the database interface providing a means to view, add and edit information about the program. Once coding is complete, a separate "annotator" SAS program could link to the database and extract information about each SAS program in a project. The annotator program would sequentially read in each SAS program and insert detailed information from the database into the header comment section. Each program, with its updated, detailed header would be written back out to a production directory for shipment to the client and archiving.

CONCLUSION

Implementation of an intranet can greatly improve existing processes and add new ones that increase department efficiency. The Analysis Programming site has become a model for other intranet sites in our organization and has facilitated communication between departments and regional offices. However, it remains a challenge for staff to find the time to supply content, even with several tools available to facilitate their contributions. Use of our intranet will increase dramatically once our standard operating procedures become incorporated into a dynamic, database-driven website that integrates coding, quality control, and project management.

Fax: (804) 951-3001

Email: WilliamsTim@PRAIntl.com

Web: www.praintl.com

REFERENCES

Bahler, C et al. 1998. SAS and HTML - HTML Publishing Using SAS. Proceedings of the Twenty-Third Annual SAS Users Group International Conference. Paper 38.

Bahler, C, S. Calhoun, E. Kistner. 1999. Essentials for Static and Dynamic Web Publishing - SAS HTML Formatting. Proceedings of the Twenty-Fourth Annual SAS Users Group International Conference. Paper 158.

Jennett, M. 1999. FrontPage 2000 Developer's Guide. Osborne/McGraw Hill. 709 p.

Kauffmann, J , K Spencer and T Willis. 2000. Beginning ASP Databases. Wrox Press. 824 p.

McConnell, S. 1996. Rapid Development: Taming Wild Software Schedules. Microsoft Press. 647 p.

Musciano, C and B Kennedy. 1998. HTML: The Definitive Guide, Second Edition. O'Reilly & Associates Inc. 531 p.

Pope, P. 1997. Using the SAS System for Large Volume HTML Document Production. Proceedings of the Twenty-Second Annual SAS Users Group International Conference. Paper 215.

SAS is a registered trademark of the SAS Institute, Inc., Cary, NC

ACKNOWLEDGMENTS

I wish to thank the Analysis Programming staff at PRA Intl. for their continuing feedback and assistance. Analysis Programmers Larry Broach, Jennifer Potter, Dev Patel and Senior Manager Lee Walke were instrumental in developing site content and layout. Special thanks to Leslie Cagley, PRA Intranet Webmaster, for assisting with site setup and always having an answer for my countless questions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Tim Williams
PRA International
4105 Lewis and Clarke Drive
Charlottesville, VA 22902
Work Phone: (804) 951-3504