Using Server-Side Include Commands for Subject Web-Page Management: An Alternative to Database-Driven Technologies for the Smaller Academic Library

Lori Northrup, Ed Cherry, and Della Darby

Frustrated by the time-consuming process of updating subject Web pages, librarians at Samford University Library (SUL) developed a process for streamlining updates using Server-Side Include (SSI) commands. They created text files on the library server that corresponded to each of 143 online resources. Include commands within the HTML document for each subject page refer to these text files, which are pulled into the page as it loads on the user's browser. For the user, the process is seamless. For librarians, time spent in updating Web pages is greatly reduced; changes to text files on the server result in simultaneous changes to the edited resources across the library's Web site. For small libraries with limited online resources, this process may provide an elegant solution to an ongoing problem.

The migration of printed subject guides and pathfinders to Web pages began almost concurrently with the creation of library Web sites. Dunsmore relates that this online migration during the 1990s was followed almost immediately by articles on the design, construction, usability, and maintenance of Web-based subject guides.[1] A scan of recent literature (for example, Dean; Roberts; Davidson; Grimes and Morris; and Galván-Estrada) suggests that online access to library resources has become the norm, and that librarians struggle with the time necessary to maintain these guides online.[2] In an effort to reduce time spent maintaining subject guides to Internet, print, and online resources, librarians are discovering more efficient methods of resource management for their Web-mounted subject guides and pathfinders.

Roberts, Davidson, and Galván-Estrada described variations of database-driven technology that generate dynamic subject guides to library resources; a common database of resources that have been descriptively enhanced for retrieval provides the backbone for a system that creates subject guides for the user at the point of query.[3] Patrons can then search for materials that may cross disciplinary lines, and receive a more targeted result list than they would have received had they only combined two static subject bibliographies from related fields. The primary advantage to this type of retrieval system, for the librarian, is that updates can be done at one central location—within the database—and will then appear when the updated item is viewed on any portion of the library's Web site generated from that database.

Libraries that have adopted database-driven technologies have done so because their resource listings have exceeded manageable capacity. Selected resources from the Internet or from library electronic holdings have reached a number that is difficult for available staff to maintain, especially across hundreds of static Web pages. For one library, this might mean a collection of more than two hundred; for larger library, this critical point might be reached only after eight hundred resources were gathered. At some point in the collection of resources, the amount of work necessary to create a database-driven system will be less than the potential workload for updating individual pages.

For the creators of these database-driven systems of resource retrieval and display, the time invested in the database system greatly outweighs the potential time lost in updating what Davidson has described as "hundreds of pages of HTML containing multiple occurrences of the same information, each of which needs to be checked and updated in response to even trivial changes in title or URL."[4] However, the investment of time and labor necessary to the creation of a populated database should not be downplayed. Davidson also notes "the process of recreating or migrating an entire site to a database-driven platform is time- and labor-intensive."[5] Indeed, Roberts suggests that the labor and time required of library staff to get the system running at full potential outweighs any technical issues involved in creating the database.[6] In Galván-Estrada's case, librarians created tools specific to the system to facilitate entry of database information, thereby increasing initial time and labor investments.[7]

This method for handling a multiplicity of Web pages and resources may work well for colleges and universities with hundreds of resources to be repeated across searches or subject pages. For some libraries, however, the critical amount of resources that can spur that type of decision may never be reached. A closer look at one small academic library's efforts to reduce time spent in updating static HTML subject guides may be helpful to other libraries in similar situations.

Samford's Situation

Samford University is a small- to medium-sized institution, with about 2,900 undergraduate and 1,500 graduate students. The campus has five

Lori Northrup (lanorthr@samford.edu) is Reference Librarian, **Ed Cherry** (cecherry@samford.edu) is the Automation Librarian, and **Della Darby** (dhdarby@samford.edu) is the Coordinator of Reference and Government Documents at Samford University Library, Birmingham, Alabama.

information units: the university library, the law library, the education curriculum center, the career development center, and the drug information center. The university library is the primary information center for all disciplines except law. Within the reference department, which is one of several university library departments, four full-time librarians are responsible for general reference, government documents reference, and maintenance of the department's designated Web pages on the library's Web site.

Like the majority of academic libraries, Samford University Library (SUL) provides subject access to its electronic resources. SUL's practice is to provide static-subject Web pages that include a list made up of (1) periodical databases of primary subject importance, databases of secondary importance, and general databases; (2) reference books; and (3) Web sites. Each reference librarian is responsible for the creation and maintenance of selected subject pages and departmental pages. The department also maintains a page with an alphabetical list of all SUL subscription databases and some free databases. It is from this list of commonly used resources that librarians select materials to fill the top portion of the subject pages.

Subject pages have undergone several metamorphoses in recent years. One major change was to add brief descriptions to each title on the database list while maintaining links to pages that held more in-depth information about each of the databases. Because a database may be listed on two or more subject pages, these changes required a considerable amount of repetitious updating. The initial changes were made to the alphabetical list, and then the HTML code was copied into each of the subject pages where the title was listed. Copying and pasting in this way helped to ensure consistency across subject pages.

When a recent review of SUL's Web-site statistics indicated that the description pages were receiving little or no activity, the reference librarians decided to enlarge the descriptions under each database title on the list and remove the links to the description pages. This would provide more initial information for the patron while eliminating the Web team's maintenance of underutilized pages. Making these changes across all the SUL subject pages and some other affected pages resulted in hours of HTML correction. For each change made to the alphabetical list of resources, several other subject Web pages had to undergo the same change. For example, librarians were copying the information for Academic Search Elite (ASE) and InfoTrac OneFile across every subject page.

Recently, faced with another update of all of the subject pages due to a database name change, the reference librarians decided to find a method that would minimize the repetition of effort required for this and future overhauls of the SUL database lists. Some informal discussion among reference librarians and with the automation librarian had concerned recent literature on database-driven Web sites; however, the general consensus was that SUL had neither the time nor the need to take such a significant leap. The total number of items on the alphabetical list of resources at that time was 144. Adopting a new platform of operation in a database-driven model seemed too large an undertaking for SUL's small list of resources. The reference coordinator consulted with the automation librarian about the possibility of using *include statements* in place of each database title and description. The Web team was already using include statements for the headers and footers of all library Web pages.

Include Commands

Include commands are a type of SSI code. Fagan explains that through using SSI codes, a Web author with no knowledge of programming can insert set groups of data into an HTML Web page. Carefully constructed statements within the HTML page give the server a command to locate and insert a piece of information (a date, a text file, a program). When a user requests a page (clicks on a link for the page), the server loads the page, inserting the requested material in place of the SSI code on the page.[8]

The Web server executes SSI code before the Web page is transferred to the browser making the request. This means that the use of SSI is not browser-dependent. SSI directives work regardless of security or privacy controls in the browser, such as disabling JavaScript or cookies.

Mach notes that this type of retrieval and substitution can be especially helpful for material that is used repeatedly on several library Web pages. The Web author can create one file containing the information used on several pages; then, when a change is made to that file, it is repeated across all Web pages that include that file.[9] As mentioned above, SUL Web pages all include the same header and footer; these headers and footers do not appear in the original HTML code for the pages. Instead, there is a command that tells the server to locate the header and footer files and include them when the page is loaded on a browser. The user does not know that SSI coding was used; for the user, the page appears complete. If a change needs to be made in a footer, then the change is made to the file containing that information. Changes appear on all pages that refer to the edited footer file.

Library Literature on the Use of SSI

Somewhat slow to adopt SSI capabilities, librarians have recently made excellent use of this elegant resource.

Current articles tend to be written with the understanding that many librarians who work on their library's Web site may not have the experience or levels of access necessary to make server changes. Articles focus more on the implementation of the syntax at the level of Web-page maintenance and less on server technology. Authors in the most recent publications have been more likely to offer examples, and to speculate on uses that, while they are not entirely innovative, have not been fully taken advantage of in the past. Using SSI to include text files in a Web page has been mentioned as a possibility since the first article, but later articles have tended to amplify the possibilities for this type of *include* command. In a slight shift of emphasis, later articles have tended to downplay the added server strain that was once a matter of great concern for SSI users.

SSI statements have probably been in use in libraries since servers were capable of processing them; documentation accompanying servers includes information on their use, as do some HTML manuals for beginning Web-page creators. Reference to the use of SSIs in a library environment does not appear until 2000. Notess's article provides a good, concise overview, and points to several helpful resources for the webmaster. Notess mentions common uses of SSI commands that will provide knowledge about the page, such as "current date and time, the LTRL of the page, the directory in which the file is located, the kind of browser the user has, or pull in content from separate files to construct the page before it is delivered." He elaborates on this point by stating that "a simple text file can contain the content [for a Web page], and people with no HTML experience can be given access to change that text."[10] While he mentions this possible use of the SSI *include* command for text files, Notess's examples all concern the echo command. Notess does mention that SSI can possibly cause server strain but states that "*includes* add very little extra load" [emphasis added].[11]

Later articles tend to mirror Notess's in describing situations appropriate for the use of SSI and in noting possible difficulties and concerns. Mach's article assumes a bit more knowledge of servers and also access to server set-up features, but it is written in clear explanatory language and provides excellent examples of commonly used SSI commands, including figures to illustrate those uses. She elaborates on Notess's suggestion about text files, and makes the suggestion that if non-HTML files are the targets of SSI commands, that they be given an extension such as *.txt,* so that they will not be indexed as Web pages.[12] Also notable is her assertion that "most Web servers should be able to handle the extra load of parsing all files and simply using the .html extension already in place."[13]

Written with an eye to those who are not Web or server administrators, Fagan's article one year later provides many example screen shots and a step-by-step guide to implementing SSI. The information here is similar to that in Mach's article, but the style is more accessible to the less-experienced Web site developer. Like Mach and Notess, Fagan mentions the use of non-HTML files and clarifies the idea of having an "all-text file, which could be ftp'd to the Web server."[14] As with the other authors, she mentions that enabling SSI on a server can result in the loss of "at least microseconds of time" as pages are parsed and reconstructed for the browser.[15] She explained, though, that "SSI is used on large Web sites in some fairly complex ways without causing any discernible time lag. The question is one of how busy your Web server is; if it is not overburdened with requests, it will easily handle the additional load of parsing files."[16] She rightly asserts that slow Internet connections or older, slower patron PCs are more likely to cause delays than is SSI load on a server.[17]

These articles demonstrate an organic movement toward wider use of SSI capabilities. All of them mention the potential to use SSI for portions of pages that are the same, such as headers and footers. Each of them also mentions the use of SSI to include text files that Web authors without HTML experience can edit. What they do not cover is the use of SSI to include text files that are repeatedly used in the body of multiple Web pages. Examples given are for Web pages with significant blocks of text that might be written at different times by different authors. The use of SSI *include* commands at SUL to insert pieces of identical text across many pages, while not groundbreaking, is significant in light of libraries' recent efforts to handle large numbers of electronic resources.

SUL's Experience

As the literature and server technology have advanced over the last few years, there has been a noticeable move from emphasis on the strain that SSI can cause to acknowledgement that the technology has become more capable in handling that strain. This is not to say that strain on the server does not occur. The user manual for Apache servers clearly states that "while this load increase is minor, in a shared server environment it can become significant."[18] However, the increases in server speed and capability certainly make it more feasible to use SSI in quantity than it was in the past, and library literature supports that view. Informal information from Web discussions and Web-development sites is much more emphatic. Recent discussion on theList at evolt.org focused on this issue and elicited the following from Baratta: "Today's servers are super charged compared to just a few years ago, and most people won't see the traffic levels that give SSI overhead a chance to affect serving time."[19]

The server statistics at SUL bear out this theory. The library's six-year-old Apache server last year averaged slightly more than 300,000 hits per month. The average increases somewhat to around 340,000 hits during the nine months of Samford's fall and spring semesters. This total number of hits includes all catalog searches as well, as this server also hosts SUL's automated library system. Despite this seemingly large number of hits, the server usually reports that it is 80 percent to 90 percent idle even during the 8 A.M. to 5 P.M. time slot during which it does the major portion of its work. At the time these statistics were gathered, SSI was already in use for the inclusion of URLs, last update dates, and some header and footer information. The commands directing the server to provide this information to the browser are more taxing to the server than commands employing the include command. After the addition of four to ten *include* commands on each of thirty-seven subject pages, and the construction of one alphabetical list consisting only of 143 *include* commands, the load on the server remains at less than 20 percent. This load is on a server with a 200 MHz PowerPC processor, with 512 MB of RAM. Certainly each library has its own server situation, and this solution for handling occasional updates to electronic resources information on library Web pages may not work for all. Consultations with the automation librarian or computer technology department should verify specific limitations and requirements for using SSI.

SUL also tested to see whether the use of SSI would result in longer page load times. The download of two identical (to the user) pages, one static and the other built from include directives, was timed. There was no visible difference between the two. A more rigorous test may have detected a difference measurable in milliseconds. However, SUL librarians feel that the user's connection speed has a larger effect on the page load time than the use of SSI.

Background

The evolution of this process for SUL moved rather quickly from inception to application, involving only slight changes to the library's normal processes for updating Web pages. The most time-consuming tasks involved creating the *include* .txt files at the outset, and then providing a database of the completed *include* commands.

Each time a database title is added to the collection, the automation librarian creates a Persistent Uniform Resource Locator (PURL) for that database. This is due to the tendency of the database vendors to alter the URL through which access is gained and because the library sometimes switches vendors for its databases. With this procedure in place, the automation librarian can make the change to the URL in one place and every link to that database will connect properly. This saves considerable time and effort for the librarians who maintain the subject pages.

Librarians were in the habit of using that PURL to create a section of HTML that presented the database link to the patron. Figure 1 illustrates a portion of HTML code for EBSCO's ASE as it appears in a text editor.

In addition to appearing in subject pages on various topics, subscription databases and commonly used Web resources are listed in one alphabetical listing. This page, updated first when changes are made, is where librarians came to copy the current code for a database. In this way, SUL attempted to keep pages consistent. Text for a database could be copied and pasted into the HTML editor as a librarian worked to update the pages.

While this seemed like a smooth process, the simple reality of having to copy and paste one database change to, potentially, all subject pages was too time-consuming to make it an efficient process. Updates to the subject pages and alphabetical list are done as time permits. Often, one or two librarians were unable to make the changes until weeks had passed, resulting in a Web site that was inconsistent and sometimes misleading. For instance, dates of database coverage might change and be documented on the alphabetical list, but not on all the subject pages listing that database.

The Way SUL Does It Now

When it was decided to try the *include* commands as an alternative for updating database information and streamlining the update process for librarians, changes were made over the course of three days, and occurred in three phases.

In phase one, the alphabetical list was carved into 143 individual .txt files, each of which contained information for one database or Web resource. Each .txt file was saved on the server with a name corresponding to the PURL that is used for that database, or with an easily recognizable name constructed from the title or URL of the resource. The HTML code in figure 1 became a .txt file titled *eb-ase.txt.* Using the PURL for the title of the file makes it more easily recognizable and acquaints librarians with the PURLs already in use.

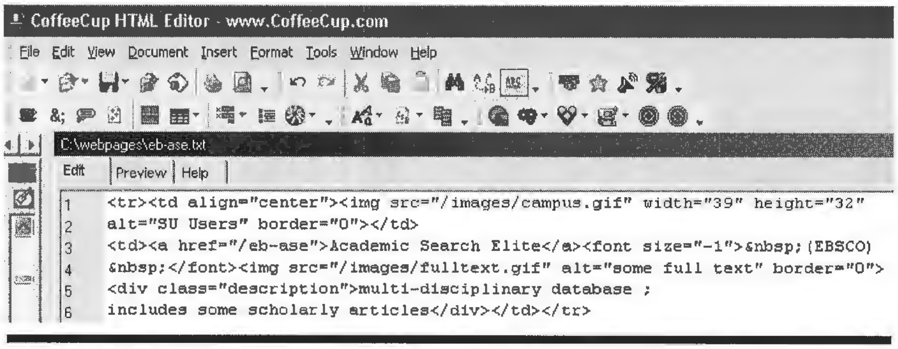
In phase two, one librarian created an Excel file containing the following: (1) an alphabetical list of resource names; (2) existing names for the corresponding .txt files; and (3) component parts of an include command for each resource. The final column concatenates these component parts together resulting in a ready-made *include* command for each resource. Figure 2 illustrates a portion of this table.

In phase three, librarians used the table of *include* commands to select the resources they wanted on their pages. This entailed replacing a block of HTML code with one include command. The example of HTML code in figure 1 would be reduced to the following statement:

> <!—#include virtual="/topics/includes/eb-ase.txt" —>

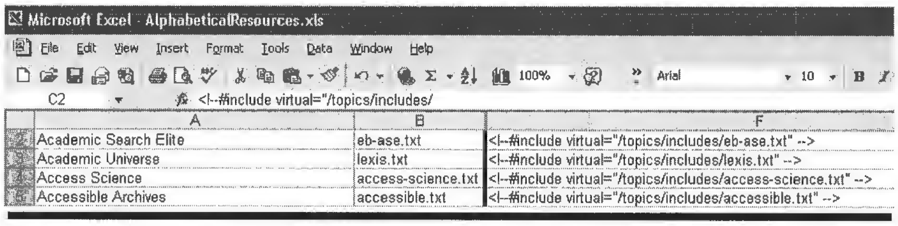
Once the appropriate pieces of HTML code had been replaced with corresponding *include* statements, the changeover was complete. From this point forward, changes in such things as database names, coverage periods, and descriptive material will be made to one .txt file. The change will immediately be reflected across all subject pages with no additional work involved for the librarians responsible for those pages.

Note that the SUL server has been configured so that it parses all Web pages. This is necessary because most of the library's Web pages have some SSI. This configuration means that the Web page extensions remain *.html*. If the server is not configured in this manner, then all pages containing SSI must end in a *.shtml* extension. This is a subject that requires discussion with automation librarians or the department responsible for the library's server.


CoffeeCup HTML Editor - www.CoffeeCup.com
File Edit View Document Insert Format Tools Window Help
C:\webpages\eb-ase.txt
Edit
Preview
Help

```
<tr><td align="center"><img src="/images/campus.gif" width="39" height="32"
alt="SU Users" border="0"></td>
<td><a href="/eb-ase">Academic Search Elite</a><font size="-1">&nbsp;(EBSCO)
&nbsp;</font><img src="/images/fulltext.gif" alt="some full text" border="0">
<div class="description">multi-disciplinary database ;
includes some scholarly articles</div></td></tr>
```

Fig. 1. HTML Code for Academic Search Elite Using a PURL Called *eb-ase*


Microsoft Excel - AlphabeticalResources.xls
File Edit View Insert Format Tools Data Window Help
100%
Arial
10

C2 <!--#include virtual="/topics/includes/

	A	B	F
2	Academic Search Elite	eb-ase.txt	<!--#include virtual="/topics/includes/eb-ase.txt" -->
3	Academic Universe	lexis.txt	<!--#include virtual="/topics/includes/lexis.txt" -->
4	Access Science	access-science.txt	<!--#include virtual="/topics/includes/access-science.txt" -->
5	Accessible Archives	accessible.txt	<!--#include virtual="/topics/includes/accessible.txt" -->

Fig. 2. Database Names, .txt File Names, and Resultant Include Commands

Advantages

Obviously, the biggest advantage to this method is the time saved for individual librarians. There is now no need for librarians to do any maintenance work for links to information housed in the alphabetical list. Static HTML pages referencing Gale's InfoTrac OneFile database, for instance, would have required updates to approximately forty subject pages; now, one librarian can correct one .txt file and simultaneously update all forty subject pages. Time saved can be used in collecting and editing the list of Web sites that are a part of each subject page; this is a task that has been pushed back in the past, in favor of making more urgent database information changes.

In addition, librarians who are using this simple technique do not need extensive training. The creation of the Excel database of *include* commands allows for quick additions to an existing page, or the creation of new subject pages. Librarians using the *include* commands can simply copy and paste them; there is no need for them to understand the syntax or to be able to repeat it. This makes using SSI particularly attractive to staff who do not want the added burden of further training in HTML. The librarian responsible for creating the .txt files and the Excel database of statements demonstrated the copying and pasting of the *include* statements to all the other librarians who edit HTML pages in a one-time ten-minute training session. The only additional training issue has involved page structure. Since the library uses a table structure for the subject pages, all table tags are included in the database .txt files. Making sure that librarians understand that they do not need to recreate the table tags has been the only additional training issue for the department.

As librarians begin to use these commands, links to resources across subject pages will look the same and will provide the user with the same information. This increased uniformity results in a more professional appearance for the Web site as a whole.

Disadvantages

This revolution in the maintenance of subject pages has not been without its disadvantages. The primary complaint by librarians using SSI *include* commands is that they cannot preview their changes in their HTML editors. SUL's department uses the CoffeeCup HTML Editor, which allows previews, but the previews are not visible for items that are retrieved using SSIs. This is because the page is not fully assembled until the server assembles it. When the librarian views the page in the editor, prior to uploading it to the server, the *include* commands are without targets. The target .txt files are on the server. When a user requests a page, *include* commands pull in the missing pieces (the .txt files, or other files); then, the completed page is seamlessly presented to the user via his or her browser. As Mach notes, "Previewing a Web page without crucial elements . . . can be disconcerting, especially to visually oriented designers."[20] In SUL's experience with this particular issue, librarians who are uncomfortable loading pages with locally invisible elements can load them into temporary folders on the server, check them for errors there, and then move them to their appropriate directories.

Conclusion

Situational factors have allowed SUL to implement this change with surprising ease and speed. Because the library has its own server, and because there is an automation librarian on staff, communication and change have been easy and efficient. Library staff deduce that it is because the *include* command of SSI is being used more than other possible commands that the library is not experiencing an increase in loading time on its pages. Of course, the size of SUL's resource list makes this kind of solution feasible; certainly, if the library were working with hundreds of resources, it would be more likely that a database-driven strategy would be adopted. The simplicity and elegance of the SSI *include* command process has encouraged adoption, and SUL has seen no ill effects from the user side of operations. Librarian Web authors quickly overcame any slight discomfort with the new process and are now able to devote a portion of editing time to other, less monotonous tasks.

References and Notes

1. Carla Dunsmore, "A Qualitative Study of Web-Mounted Pathfinders Created by Academic Business Libraries," *Libri* 52, no. 3 (Sept. 2002): 140–41.
2. Charles W. Dean, "The Public Electronic Library: Web-based Subject Guides," *Library Hi Tech* 16, no. 3–4 (1998): 80–88; Gary Roberts, "Designing a Database-Driven Web Site, or, The Evolution of the InfoIguana," *Computers in Libraries* 20, no. 9 (Oct. 2000): 26–32; Bryan H. Davidson, "Database-Driven, Dynamic Content Delivery: Providing and Managing Access to Online Resources Using Microsoft Access and Active Server Pages," *OCLC Systems and Services* 17, no. 1 (2001): 34–42; Marybeth Grimes and Sara E. Morris, "A Comparison of Academic Libraries' Webliographies," Internet *Reference Services Quarterly* 5, no. 4 (2001): 69–77; Laura Galván-Estrada, "Moving towards a User-Centered, Database-Driven Web Site at the UCSD Libraries," *Internet Reference Services Quarterly* 7, no. 1–2 (2002): 49–61.
3. Roberts, "InfoIguana"; Davidson, "Database Driven"; Galván-Estrada, "User-Centered, Database-Driven Web Site."
4. Davidson, "Database Driven," under "Introduction."
5. Ibid., under "Development Considerations."
6. Roberts, "InfoIguana," 32.
7. Galván-Estrada, "User-Centered, Database-Driven Web Site," 55–56.
8. Jody Condit Fagan, "Server-Side Includes Made Simple," *The Electronic Library* 20, no. 5 (2002): 382–83.
9. Michelle Mach, "The Service of Server-Side Includes," *Information Technology and Libraries* 20, no. 4 (2001): 213.
10. Greg R. Notess, "Server Side Includes for Site Management," *Online* 24, no. 4 (July 2000): 78, 80.
11. Ibid.
12. Mach, "Service of Server-Side Includes," 216.
13. Ibid., 214.
14. Fagan, "Server-Side Includes Made Simple," 387.
15. Ibid., 383.
16. Ibid.
17. Ibid.
18. Apache HTTPD Server Project, "Apache HTTP Server Version 1.3: Security Tips for Server Configuration," The Apache Software Foundation. Accessed Oct. 29, 2003, http://httpd.apache.org/docs/misc/security_tips.html.
19. Anthony Baratta, e-mail to theList mailing list, May 16, 2003, Accessed Nov. 4, 2003, http://lists.evolt.org/archive/Week-of-Mon-20030512/140824.html.
20. Mach, "Service of Server-Side Includes," 217.

Index to Advertisers

Art & Tech	179	LITA	cover 2, 191, covers 3–4
EBSCO	200		