

Porting Data Collected With Custom Internet Forms to SAS

Robert Workman
Computer Science Department
Southern Connecticut State University workman@scsu.ctstateu.edu

David Workman
Brazil Center, Institute of Latin American Studies
University of Texas at Austin dworkman@mail.utexas.edu

Abstract

This presentation discusses the use of the Perl programming language to transform data collected from Internet HTML forms into SAS readable format. Detailed examples of the techniques used are taken from work done for the Brazil Center at the University of Texas.

1. Background

The purpose of the project is to create SAS readable output from Internet data forms. We illustrate our methods for accomplishing this with an example from the Brazil Center at the University of Texas. The application provides a good example of the power of the Internet as a communications tool. People interested in Brazil Center activities are dispersed throughout the world. They tend to be at universities or agencies that have access to the Internet. The data that the Brazil Center needs is time sensitive. The volume of data is large enough that SAS is an appropriate tool for analysis. We feel that with the ability to convert Internet forms answers into SAS readable format will lead many faculty and students to develop research uses based on this application.

Our solution to creating SAS readable files from the responses to questions on Internet forms involves working with Hypertext Markup Language (HTML), a Perl language Common Gateway Interface (CGI) program, and the UNIX operating system. Our objective is to keep things as simple as possible while developing a model application that can be expanded upon.

Table 1 shows the name of the files that we worked with, their use, the UNIX directories that they were placed in, their UNIX permission level, and their use. Figure 1 is a print out of the form that was created. The files that were created for this project will be described.

Table 1: File names, locations, properties, and use

File, Internet Page, Directories	Printout	File Name	Directory	UNIX Permission Level	Use
Internet Page	Figure 1				The user sees and responds to this page.
HTML code	Figure 2	bc.html	public_html	644	Code that created the Internet page.
Perl code	Figure 3	bc.cgi	public_html/cgi_bin	755	Code converts user responses into SAS readable data.
Perl code output	Figure 4	bc.txt	public_html/cgi_bin	666	SAS readable data, with space delimited fields.
UNIX Directories				755	Required for Internet use.

2. The Internet Page (see Figure 1).

Figure 1 shows the page that the user sees when our web site is selected. The Internet data form is only a demonstration that was developed to illustrate several types of inputs that are available on web pages. Shown are text, radio button, and dropdown lists. Checkbox inputs, which allow the user to select more than one option, were replaced by a series of radio buttons to insure that each possible response has a specific yes or no answer. The response for the radio buttons is indicated by either checking the degrees obtained or by leaving the default selection unchanged, if the user does not have a degree. The dropdown list, which displays various departments, is activated by clicking on the down arrow in the "What department are you most interested in?" box. The form also includes a submit button, labeled "Send Form". Clicking on this button sends the user's responses to our web site to be processed. There is also a clear form button.

Figure 1: Internet Page

Brazil Center Information Form

Please enter your name.

First Name: Last Name:

What degrees do you have?

☐ Bachelor of Arts

☐ Bachelor of Sciences

☐ Master of Public Affairs

☐ Other Degrees

What department are you most interested in?

3. HTML code (see Figure 2).

The web browser, most likely Netscape, uses the HTML code listed in Figure 2 to display the web page shown in Figure 2. Some comments about the forms related lines of code follow. Of particular note is the line:

```
<form action="http://www.scsu-cs.ctstateu.edu/cgi-bin/workman/bc.cgi" method="get">
```

The "form action" attribute informs the browser that when the Submit button is clicked the user responses are to be sent to the server and a specified program is to be executed using the responses. In this case, the server is located at Southern Connecticut State University and is named 'scsu-cs.ctstateu.edu'. The program to be executed is a cgi script named "bc.cgi". The "method" attribute is used to designate the method by which the browser sends data to the server. The "get" designation contacts the server and sends the data to the site specified in the action attribute in a single transmission step. The "get" method is recommended for small forms; larger forms require the use of the "post" method.

Figure 2: HTML Code

```
<HTML>
<TITLE>Brazil Center Information Form</TITLE>
<BODY>
<H1>Brazil Center Information Form</H1>
<FORM
action="http://www.scsu-cs.ctstateu.edu/cgi-bin/workman/bc.cgi"
method="GET"> <P>
<!-- Text input - The user enters text. -->
<b>Please enter your name.</b><p>

First Name: <input NAME="firstname">
Last Name: <input NAME="lastname">
<P>
<b>What is your status at UT Austin? </b>
<!-- Radio Button Input - The user may select only one option.-->
<P>
<input type = "radio" name="status" value="faculty">Faculty
<input type = "radio" name="status" value="vscholar">Visiting Scholar
<input type = "radio" name="status" value="other" checked>Other<p>

<b> What degrees do you have? </b><p>
<!-- The user may select any combination of options. Radio buttons are used
instead of checkboxes to create an answer for each option.-->

Bachelor of Arts
<input type="radio" name="degreeBA" value="BA"> Yes
<input type="radio" name="degreeBA" value="no" checked> No <p>
Bachelor of Science
<input type="radio" name="degreeBS" value="BS"> Yes
<input type="radio" name="degreeBS" value="no" checked> No <p>
Master of Arts
<input type="radio" name="degreeMA" value="MA"> Yes
<input type="radio" name="degreeMA" value="no" checked> No <p>
Other Degrees
<input type="radio" name="degreeOther" value="Other"> Yes
<input type="radio" name="degreeOther" value="no" checked> No <p>

<P>
What department are you most interested in?<p>
<select name ="department">
    <OPTION>Spanish/Portuguese
    <OPTION>PublicAffairs
    <OPTION>Government
    <OPTION>Other
</select>
<p>
<input TYPE="submit" VALUE="Send Form Data">
<input TYPE="reset" VALUE="Clear Form Data">
</FORM>
</BODY>
</HTML>
```

4. Perl code (see Figure 3)

Perl is a programming language designed for manipulating text, files, and processes in the UNIX environment. A library of subroutines for working with Internet data, "cgi-lib.pl", was created by Steven Brenner. The line of code: *require "cgi-lib.pl"* tells Perl to use that library. The code *&ReadParse* causes a cgi-lib routine to execute. This routine, "ReadParse" converts data sent by Internet into text and places the data in an associative array name "in". Data in an associative array can then be accessed by field name. The line of code that starts, *print OUT \$in{firstname}, " ", \$in{lastname}...*, illustrates the use of an associative array "in". Firstname is the name of a field specified in the HTML program. *\$in{firstname}* is the contents of the field, for example Marcia. The "print" instruction sends the contents of the associative array "in" to the file designated in the line of code: *\$myfile = ">>bc.txt"*. The double greater than signs mean append data to the file.

Figure 3. Perl Code file, bc.cgi

(The # character indicates a comment.)

```
#!/usr/local/bin/perl
$myFILE = ">>bc.txt"; #append data to file bc.txt
require "cgi-lib.pl";
MAIN:
{
    print "Content type: text/html\n\n";
    &ReadParse; #Places input data in an associative array named in;
    print $in;

    open(OUT,$myFILE);
    #The associative array in contains user responses.
    #The following statement sends the responses to the file bc.txt
    # $in{"firstname"} will contain a name such as Jim.
    # $in{"lastname"} will contain a name such as Smith.
    # " " places a blank after each field.
    # If fields need to be separated by a , use ","
    print OUT $in{"firstname"}," ",
        $in{"lastname"}," ",
        $in{"status"}," ",
        $in{"degreeBA"}," ",
        $in{"degreeBS"}," ",
        $in{"degreeMA"}," ",
        $in{"degreeOther"}," ",
        $in{"department"}," ",
    close(OUT);
    exit 0;
}
```

The example form presented in this paper stresses simplicity. Perl, however, is a powerful text manipulation language. For example, special characters can be checked for and removed, and input can be checked for validity. A feature of Perl is that there are many existing resources to facilitate the work of people who do not want to do more advanced Perl coding. The text published by O'Reilly & Associates, listed below, provided good introductory and reference material. *Introduction to CGI/Perl*, by Steven Brenner, focuses on creating web scripts. There is

also a USENET newsgroup (comp.lang.perl) devoted to Perl language coding and Perl related listservers, such as 'perl-win32-users@mail.hip.com'.

5. Perl code output (see Figure 4)

As can be seen from this sample output, a space delimited ASCII text file has been created. Perl code can be written to reformat the data if this is required.

Figure 4. Sample Perl code output, SAS input

Jim Smith vscholar BA no MA no Spanish/Portuguese Al Jones faculty BA no no Other PublicAffairs Mary Green vscholar no BS MA no Spanish/Portuguese
--

Note: Jim Smith is a visiting scholar, has a BA and an MA degree, but no MS or other degree. He is interested in the Spanish/Portuguese department.

6. UNIX issues

It comes as a shock that while a 1984 reference book is adequate for working with UNIX a 1996 book for working with Netscape is out of date. We were fortunate to have people around who both help us get started and tolerated our complaining. We would like to thank Matt LeSieur, a Computer Science major at Southern Connecticut State University, and Bharat Narumanchie, a computer engineering major at Carnegie-Mellon for their help. We have included in Table 1 directory requirements for the HTML code and CGI program. We have also included UNIX Permission Levels that work. If these permission levels are set incorrectly, confusing or no error messages are given. For example, if the output file is not given write permission the result is no error message, but there is as no data in the file.

7. Summary and Conclusion

The combination of HTML, Perl and UNIX applications discussed in this paper provides an example of a relatively straight forward method for integrating Internet data collection and SAS. As the use of the Internet increases and data bases become larger and more complex, the usefulness of SAS as a tool for analysis will also increase. The example presented in the paper can readily be expanded upon to include a greater variety of HTML input formats and to encompass a broader category of variables.

References

HTML The Definitive Guide, Chuck Musciano and Bill Kennedy, O'Reilly & Associates, Inc. Sebastopol, CA, 1996.

Introduction to CGI/Perl, Steven E. Brenner and Edwin Aoki, M&T Books, New York, NY, 1996.

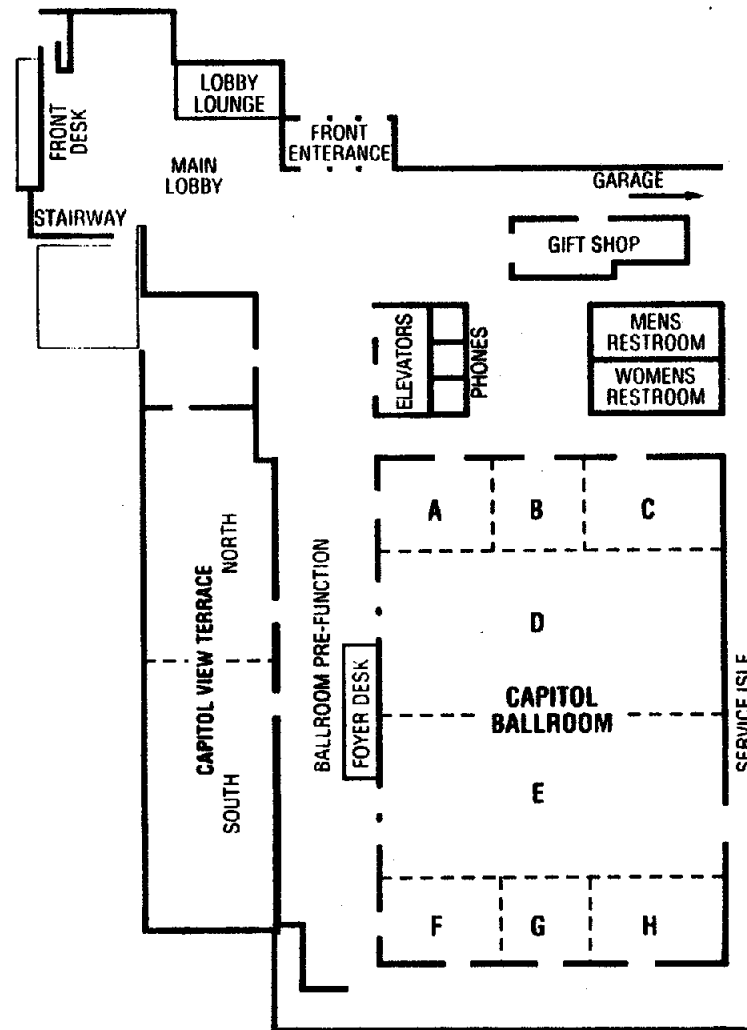
Learning Perl, Randal L. Schwartz, O'Reilly & Associates, Inc. Sebastopol, CA, 1993.

Real World UNIX, John D. Halamka, Sybex Computer Books, Berkeley, CA, 1984.

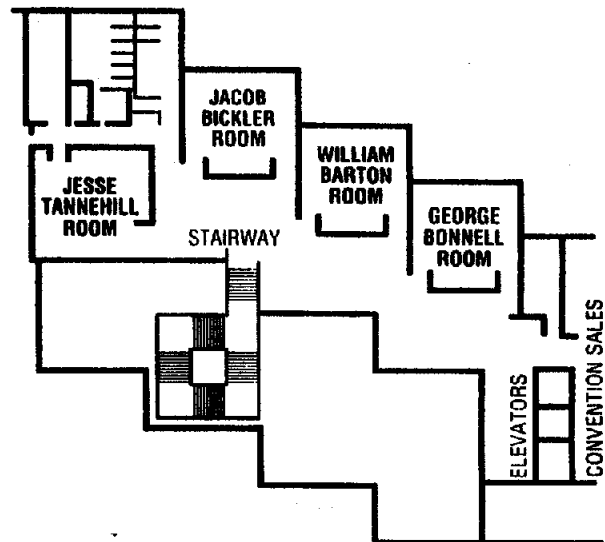
Austin Marriott at the Capitol

Austin, Texas

3rd Floor Main Entrance



4th Floor



ANNOUNCEMENT & CALL FOR PAPERS

Seventh Annual
South-Central Regional SAS Users' Group Conference
Houston Marriott Westside, Houston, Texas
November 9-11, 1997



This two-day, regional conference for SAS users from Texas, Louisiana, Oklahoma, Arkansas, New Mexico and adjoining states will include presentations of technical papers, poster presentations, roundtable luncheon discussion meetings, demonstrations, workshops, "birds-of-a-feather" sessions, problem-solving sessions, and one-on-one technical exchanges. The conference will feature general sessions for all participants, as well as concurrent sessions covering various areas of specialization.

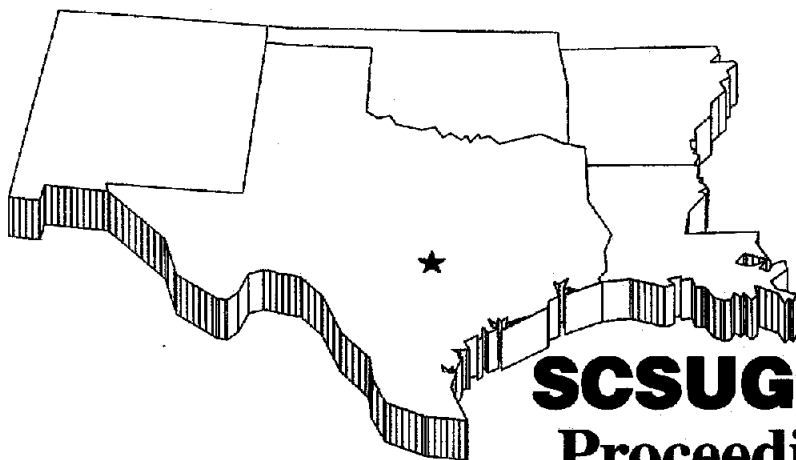
Please consider sharing your methods, experiences and results with other conference attendees. We anticipate that papers will be presented in the following sections: Applications Development, Coders' Corner, Forecasting, Future Products, Statistical Computing, Training, and Tutorials. Papers covering topics in these, and other, areas would be most welcome. There will be a published conference proceedings.

Proposed papers, or abstracts of papers, should be sent to Tom Winn, 10906 Rustic Manor Lane, Austin, TX 78750, by July 15.

Volunteers are needed to help with registration/check-in, and as roundtable discussion leaders, messengers, session coordinators, timekeepers, and other facilitators at the conference.

If you are able to help, please contact Conference Chairperson Deborah Babcock Buck by telephoning (713) 855-1027, via postal mail at D.B. & P. Associates, 7310 Starbridge Drive, Houston, TX 77095, or via the Internet at 103525.2762@compuserve.com.

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



SCSUG '96
Proceedings