

Lab Assignment #3

Description

This lab assignment is graded on the scale of **credit** / **no-credit**, and you may work in groups of 2. Please submit Lab #3 online. Remember to include the user name of each teammate in the correct format (on separate lines) as specified by the README-submit.

This lab assignment is due Wednesday, August 4, 2004 at 11:59pm. It should not take too long, and is meant to help you understand HW #3. Put all answers to questions in your README file. Include in your submit directory the source code of all programs written.

Set up CGI on Leland

If you haven't already, each teammate should go to <http://cgi.stanford.edu/> and set up his/her personal CGI capability. This will automatically install a cgi-bin in your leland account, with a default Perl CGI program and a PHP page. It may take a day or two to get set up, so do this as soon as possible.

Hypertext Markup Language (HTML)

One of the best ways to learn how to work in different programming/markup languages is to write lots of code! Here we will hand-code an HTML form.

1. Open up a file called `guestbook.html`, and save it in your web space (i.e. somewhere under `~/user/www/`).
2. Add a title tag so that your guest book shows a title on the browser's title bar: "My Guest Book!"
3. Add a form to your HTML document. This form should use the GET method.
4. Add a textarea to your form. It should have 80 columns, 20 rows, and should be named "MyEntry".
5. Add a submit button to your form. It should be named "MySubmit".
6. Now, load this HTML page by pointing your browser to the URL:
`http://www.stanford.edu/~user/guestbook.html`
7. Type a paragraph of something into the text area, and submit the form. Notice that the URL is super long. This is because we used the GET request method, and all variable value bindings are passed via the URL. If we had used the POST request method, this would not be the case.

[Question 1: Text sent in the variable=value bindings of an HTML Form with method=GET are all URL encoded. What symbol is the space character encoded as? What is CRLF encoded as?]

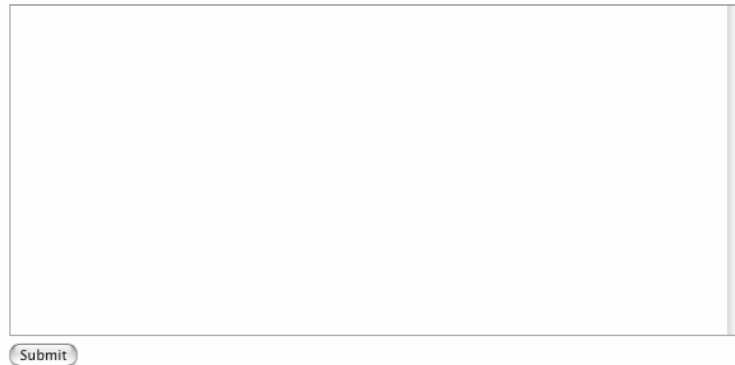


Figure 1. HTML Form in `guestbook.html`

The Power (and Danger) of CGI Programming

Thus far, you have seen examples where CGI programs are just Perl programs that reside in your `cgi-bin` directory. The power of CGI extends far beyond just that. First, CGI programs can be arbitrary executable programs! They can be an executable shell program, a Perl or Python script, or even a compiled C/C++ program!

Let's write a CGI program in C. Go to your `cgi-bin` directory, and open a new file called `guestbook.c`. Write a standard helloworld program in C. Then compile it by typing

```
gcc -o guestbook guestbook.c
```

 Run the program by typing `./guestbook`.

Since `guestbook` is an executable file, you can access it by pointing your browser to `http://cgi.stanford.edu/~user/guestbook`. Oops.... What happened? You get a Server error, with an error message that says you have a bad header. To solve this, you need to think about what is the most important thing that comes back as part of the CGI HTTP response.

Go back to your `guestbook.c` file and add a `printf` that prints out the Content-type as `text/html` on a single line. This `printf` has to go before your other `printf`s. In fact, it has to be the first `printf` in your CGI program! Remember also that there is always a blank line between the headers and the HTTP response's body. A `printf("\n");` will solve that problem. Recompile, and refresh the page in your browser. You should see "Hello World!" sent to your browser. ☺

When `http://cgi.stanford.edu/~user/guestbook` is requested, the server runs your executable, and returns the outputs as the HTTP response.

However, it is useful to note that Perl programs (especially those written in this manner) are subject to many security issues. Skim this document:

<http://gunther.web66.com/FAQS/taintmode.html>

[Question 2: Is the Perl taint mode a compile time check, or a run-time check? Does this imply anything about the performance of taint mode? Also, are variables passed in via hidden input tags considered tainted, or are they not?]

Here is a trick: you can also run Java programs through CGI. How? Create a file in your cgi-bin directory called GuestBook.java. Paste in this source:

```
public class GuestBook {  
  
    public static void main(String[] args) {  
        System.out.println("Content-type: text/html");  
        System.out.println();  
        System.out.println(  
            "<html><body>How YOU doin' ?</body></html>");  
    }  
}
```

Compile it to GuestBook.class. Now, open up new file called GuestBook.pl. Paste in this source:

```
#!/usr/bin/perl -w  
system("java GuestBook");
```

What did we just do here? Basically, you have a Perl CGI program that will run a Java program. The output of any programs that this Perl CGI program invokes will also be packaged up and sent across as part of the HTTP response. If you now open <http://cgi.stanford.edu/~user/GuestBook.pl>, you will see the output of your Java program. Neat huh?

[Question 3: Edit your Java program (GuestBook.java) to read in a file and print it to standard output (System.out). The file should be the guestbook.html file that was in your WWW directory. Since the WWW directory cannot be accessed by the cgi server, you must move that guestbook.html file over to your cgi-bin file.]

Note that we have just done a very interesting thing. We have a Perl CGI program that actually runs a Java program. This Java program can do some complicated processing, and then read and write out whole html files (or snippets of html stored in files). This way, you can separate the Java coding (back end) from the HTML authoring (front end). If you were a company, you could do this to allow your web/graphic designers to design all the pages, and your coders (who don't have much aesthetic talent, unfortunately) to do all the infrastructure coding. However, orchestrating all of this can be a mess.... Fortunately, we have Java Servlets and JSP to make our lives easier.

You can also imagine how we could extend the ~20 lines of code you just wrote into a full-fledged Java/Perl/CGI guestbook for your web site.

1. You would need a file to store guest book entries (e.g. guestbookentries.txt).
2. You would need an html snippet file (like guestbook.html) to help render the input forms.
3. You would assign your form's action field to point to a Perl CGI program that would read in the QUERY_STRING, and pass it as an argument to a Java program.
4. The Java program would take in that argument, open up your guest book entries file, and append the text from the argument to your guest book entries file. Future requests to your entries file will see the newly added entry!

It's as simple as that. =) We won't ask you to do that, because it's just busy work.

Perl CGI Debugging

Read the HW #3 assignment. In the assignment, we spoke of an "sboxd" URL that allows you to get debugging output. Unfortunately it seems the ITSS people have removed the sboxd URL. They might put it up again, but until that happens, you can add the line:

```
use CGI::Carp qw(fatalsToBrowser);
```

to your Perl program. What this does is send all error output to the browser. Here, I have a helloworld CGI program, with an obvious error. Try running it with/without the Carp import.

```
#!/usr/bin/perl -w
use CGI::Carp qw(fatalsToBrowser);
use strict 'vars';

my($EOL) = "\015\012";
my($nowStr);
$nowStr = localtime();

xxx THIS IS AN OBVIOUS ERRORxxxx

print "Content-type: text/html$EOL$EOL";

print "<html><head><title>Hello.pl</title></head>\n";
print "<body bgcolor=white>\n";
print "<h1>Hello.pl</h1>\n";
print "Hello there from CGI-land. It's currently '$nowStr'\n";
print "</body></html>\n";
```

Here are two links that discuss how to debug Perl/CGI programs. Skim them.

http://faq.his.com/web2/Plesk/webhosting_debug_perl_basic.html

http://faq.his.com/web2/Plesk/webhosting_debug_perl_advanced.html

HTML Page Prototyping

You will need to output some HTML forms for HW #3. Unfortunately, it is usually difficult to write HTML straight into your CGI program without too many bugs, because the tags are usually interspersed within programming logic. That's why we will write the HTML pages separate from the program, and then copy and paste relevant parts into our CGI program as necessary.

[Question 4: Read the HW #3 assignment handout. Create a new file, `collision.html`, that will be rendered like the HTML page on page 9 of the assignment handout. Your page does not have to look exactly like it, but it should include all the form elements and text labels. Note that we have a spelling mistake on the figure on page 9. Please don't be as bad as we were...]