CGI Made Really Easy

or, Writing CGI scripts to process Web forms

Home > Web Technology Made Really Easy > CGI Made Really Easy

Donate

Go to Footnotes

April 12, 2002-- The getcgivars() routines provided below have been updated to allow ";" as well as "&" for parameter separators. If you're using an old version, please upgrade to the new one.

Auf deutsch (in German), as translated by Friedemann Wachsmuth
En español (in Spanish), as translated by René Alvarez
Em português (in Portuguese), as translated by Clay Chagas
In het Nederlands (in Dutch), as translated by Simon Amstel

So it's 4:00, your boss needs a CGI script written by 4:30, and you don't even know what CGI stands for. You've come to the right place.

There's not much to it, despite any intimidating hype you might have heard. If you can read from STDIN and write to STDOUT, then you can write CGI scripts. If you're a programmer already, this primer can teach you the basics of CGI in a few minutes. *If you're not a programmer, this primer won't help you much-- sorry. Learn some programming, even shell scripting, and come back when you're done. Good luck!*

This primer focuses on writing CGI scripts to process HTML forms on the Web. It skips some details, but can bring you up to speed *fast* (literally an hour or less), and covers 90% of real-world situations. When you feel the need, check out the full CGI spec. For help with writing HTML forms, see this tutorial fragment, or this documentation at NCSA.

Once you've read this page, see the footnotes for sample scripts and other topics.

What is CGI?

CGI is not a language. It's a simple protocol that can be used to communicate between Web forms and your program. A CGI script can be written in any language that can read STDIN, write to STDOUT, and read

environment variables, i.e. virtually any programming language, including C, Perl, or even shell scripting.

Structure of a CGI Script

Here's the typical sequence of steps for a CGI script:

1. Read the user's form input.
2. Do what you want with the data.
3. Write the HTML response to STDOUT.

The first and last steps are described below.

Reading the User's Form Input

When the user submits the form, your script receives the form data as a set of name-value pairs. The names are what you defined in the INPUT tags (or SELECT or TEXTAREA tags), and the values are whatever the user typed in or selected. (Users can also submit files with forms, but this primer doesn't cover that.)

This set of name-value pairs is given to you as one long string, which you need to parse. It's not very complicated, and there are plenty of existing routines to do it for you. Here's one in Perl, a simpler one in Perl, or one in C. For a more elaborate CGI framework, see Perl's CGI.pm module. The CGI directory at Yahoo includes many CGI routines (and pre-written scripts), in various languages.

If that's good enough for you, skip to the next section. If you'd rather do it yourself, or you're just curious, the long string is in one of these two formats:

```
"name1=value1&name2=value2&name3=value3"
"name1=value1;name2=value2;name3=value3"
```

So just split on the ampersands or semicolons, then on the equal signs. Then, do two more things to each name and value:

1. Convert all "+" characters to spaces, and
2. Convert all "%xx" sequences to the single character whose ascii value is "xx", in hex. For example, convert "%3d" to "=".

This is needed because the original long string is URL-encoded, to allow for equal signs, ampersands, and so forth in the user's

input.

So where do you get the long string? That depends on the HTTP method the form was submitted with:

- For GET submissions, it's in the environment variable QUERY_STRING.
- For POST submissions, read it from STDIN. The exact number of bytes to read is in the environment variable CONTENT_LENGTH.

(If you're wondering about the difference between GET and POST, see the footnote discussing it. Short answer: POST is more general-purpose, but GET is fine for small forms.)

Sending the Response Back to the User

First, write the line

```
Content-type: text/html
```

plus another blank line, to STDOUT. After that, write your HTML response page to STDOUT, and it will be sent to the user when your script is done. That's all there is to it.

Yes, you're generating HTML code on the fly. It's not hard; it's actually pretty straightforward. HTML was designed to be simple enough to generate this way.

If you want to send back an image or other non-HTML response, here's how to do it.

That's it. Good Luck!

See how easy it is? If you still don't believe me, go ahead and write a script. Make sure to put the file in the right place on your server, and make it executable; see this footnote for more hints.

Before you write too many scripts, learn about CGI security issues.

When you need to know more about CGI, see the complete CGI specification at NCSA. W3C also maintains a CGI page with links to any documents you might need.

Oh yeah, CGI stands for Common Gateway Interface-- in other words, a standard ("common") way of communicating ("interface") between different

processes ("gateway", sort of).

Other Things to Play With

See the Footnotes Page for the following stuff:

1. Sample CGI programs
2. CGI Mailer Script
3. Security with CGI Scripts
4. Placing Your Script on the Server
5. Sending an Existing File Back as a Response
6. Other Useful CGI Environment Variables
7. Returning an Image or Other Non-HTML Response from a CGI Script
8. What is the difference between GET and POST?
9. Gaining More Control, with Non-Parsed Header Scripts

© 1996-1998, 2002 James Marshall
(comments welcome; for questions, please scan the FAQ first)

Last Modified: April 12, 2002 *http://www.jmarshall.com/easy/cgi/*