# Lisp & CGI
# a tutorial for new programmers

Gene Michael Stover

created Sunday, 28 September 2003
updated Monay, 12 April 2004



## Contents



## 1 Readme

As of 12 April 2004, I'm updating this article. I'll try to work on it a little bit every night until it's updated.

If you're an SDF Lisp programmer, do what I've recommended in Section 3. Some of the techniques currently discussed in this article won't work on Open Lisp; I'll try to change that as I update the article this week.

## 2 Introduction

This is a tutorial for new Lisp programmers, but not total beginners. I assume you know some basic Lisp, like the syntax, lists, and what `car` & `cdr` mean.

Things I discuss or show in this tutorial include

- what CGI is,
- obtaining & parsing the inputs to a CGI program,
- producing output from a CGI program,
- another programmer's thought processes as he develops some software. It's not that my thought processes are models for a good programmer; it's that it is sometimes useful to see how another practitioner of the same craft thinks.

## 3 Things to Learn First

Before you continue reading this article, it would be a good idea to know how to run scripts written in Lisp from the unix command line. For that, see my article "Lisp Script" (**??**). If you're an SDF user & you just want to get started quickly, never fear; there is a chapter towards the end of that article which is a quick start for SDF programmers. When you have your own command-line "Hello world" Lisp script, come back here.

Possibly of interest before or after reading this tutorial is another I wrote, "Generating HTML with Lisp / a tutorial for new programmers" (**??**).

## 4 What CGI Is

CGI stands for Common Gateway Interface. It is a protocol by which a web server can communicate a web browser's request to a program so that the program can fulfill the request. The program is commonly, but not necessarily, external to the browser.

The protocol relies on environment variables to communicate the details of a browser's request to the program. Some of those environment variables are listed in Figure 1.

Though CGI is more of a de facto standard than an official one, there is a draft specification of the CGI protocol online ([3]). You can find some pretty comprehensive information about CGI at the W3C website ([2]). There is also

| | | |
|---|---|---|
| DOCUMENT_ROOT | GATEWAY_INTERFACE | HTTP_ACCEPT |
| HTTP_ACCEPT_CHARSET | HTTP_ACCEPT_ENCODING | HTTP_ACCEPT_LANGUAGE |
| HTTP_CONNECTION | HTTP_COOKIE | HTTP_HOST |
| HTTP_KEEP_ALIVE | HTTP_REFERER | HTTP_USER_AGENT |
| PATH | QUERY_STRING | REMOTE_ADDR |
| REMOTE_PORT | REQUEST_METHOD | REQUEST_URI |
| SCRIPT_FILENAME | SCRIPT_NAME | SERVER_ADDR |
| SERVER_ADMIN | SERVER_NAME | SERVER_PORT |
| SERVER_PROTOCOL | SERVER_SIGNATURE | SERVER_SOFTWARE |

Figure 1: Some of the environment variables CGI uses

some nice information on a website at the University of Illinois at Urbana-Champaign ([1]).

*fixme: Who defines CGI? Is it an official standard or a de facto standard? From where can the standard's documentation be obtained?*

# 5 Getting Started

Find a computer that will be your web server. Maybe it's your own, or maybe you rent space on it.

Ensure that a web server is installed & running on it. If you are coming at Lisp & CGI without any previous experience with CGI, you might want to use Apache because that's what I'm using for this article. If you need to install Apache from scratch, you can download it from Apache's web site, http://apache.org/.

The web server must be configured to execute CGI programs. A common practice is, or was, to put all CGI programs in a directory called `cgi-bin`, but I prefer to put them anywhere in the web server's directory tree & to ensure that their filenames end with "`.cgi`".

To configure Apache so that it considers any filename matching "`*.cgi`" as a CGI program, experiment with these edits in Apache's configuration file:

1. In the <`Directory`> section, there is an `Options` item. Make sure that `ExecCGI` is one of the arguments to `Options`.

2. Ensure there is a line like this: "`AddHandler cgi-script .cgi`". I think there is one in the default configuration file, & you just need to uncomment it.

I said to "experiment" with those changes because I've never been sure which one actually made Apache work the way I wanted. Maybe it required those & some other change; I've never known for sure. Also, Apache's configuration files change slightly on some new releases, so it is possible that the configuration file for my Apache does not use exactly the same syntax & semantics that yours does.

```
#! /bin/sh
### A simple CGI program.
### Prints a hard-coded HTML page.
echo content-type: text/html
echo ""
echo \<html\>
echo \<head\>\<title\>titular text\</title\>\<head\>
echo \<body\>
echo \<p\>I guess CGI is configured properly.\</p\>
echo \</body\>
echo \</html\>
```

Figure 2: A simple CGI program to test that your web server is configured properly

How do you know when your web server is configured to run CGI programs? There's a pretty easy test. Locate or create the directory in which you want to put some CGI programs. Double-check permissions on it. On any kind of Unix, you should be the owner, & the permissions should be 755. Then create a file called `test.cgi` & copy the code from Figure 2 into it. Make sure the permissions on `test.cgi` are 755. Then point your browser at `test.cgi`. For example, if your account name were `gene`, you might direct your browser to `http://localhost/gene/test.cgi`.

In case it's more convenient, that same tiny CGI program is in a text file online[1]. You can also see it work on my web server.

Once your web server is configured properly, you're ready to rock . . . I mean, ready to Lisp.

# 6 A Super-Simple Lisp CGI Program

Now let's write a super-simple Lisp CGI program. In the directory where you'll put CGI programs, create a file called `lisp00.cgi` & copy into it the text from Figure 3. In case it's more convenient, the program is also in an online file.

Remember to update the first line of the program so that it points to your Lisp interpreter!

You may be wondering what's happening in that program, or you may be fighting the urg to wretch at all the explicit `format` expressions, but bear with me for the moment. I'll explain everything in time. For now, make sure the program works on your web server, or see it work on mine[2].

If you have problems getting your own super simple Lisp CGI program to work, check these things:

[1] test-cgi.txt
[2] lisp00.cgi

```
#! @HOME@/bin/lisp-script
;;; -*- Mode: Lisp -*-
;;;
;;; A simple CGI program in Lisp.
;;;

(format t "Content-type: text/html~%")
(format t "~%")
(format t "<html>~%")
(format t "<head>~%")
(format t "<title>why do I like &quot;titular&quot;</title>~%")
(format t "</head>~%")
(format t "<body~%>")
(format t "<p>What is it?~%")
(format t "It's a simple CGI program~%")
(format t "in Lisp~</p>~%")
(format t "</body>~%")
(format t "</html>~%")
```

Figure 3: A very simple Lisp CGI program

1. Do you have the correct URL?
2. Is a web server running on your server? (Appendix A)
3. Do you have the correct pathname to your Lisp interpreter in the first line of your Lisp CGI program? And does that line begin with `#!`? (See [4].)
4. Are the permissions correct on your Lisp CGI program? They should be 755, which is also known as `rwxr-xr-x`.
5. Do you have correct permissions on the directories leading to your Lisp CGI program? They should be 755.
6. Is your web server configured to run CGI programs? Check this by trying some other CGI program, maybe the one in Figure 2.

# 7 Environment Variables

With CGI, the web server communicates much information about a query through environment variables.

The good thing about using environment variables is that they are a simple, reasonably portable way of communicating this type of information. Environment variables work in all Unices & even in Microthought Winders.

```
#! /usr/local/bin/clisp
;;; Make an HTML table of the CGI environment variables.

(format t "Content-type: text/html~%")
(format t "~%")
(format t "<html>~%<head>~%")
(format t "<title>CGI environment variables</title>~%")
(format t "</head>~%<body~%><table border=\"1\">~%")
(mapc #'(lambda (env)
  (format t "<tr><td>~A</td> <td>~A</td>~%"
  env (or (system::getenv env) "")))
      '("AUTH_TYPE" "CONTENT_LENGTH"
        ...
"SERVER_PROTOCOL" "SERVER_SOFTWARE"))
(format t "</table></body>~%</html>~%")
```

Figure 4: A very simple Lisp CGI program

The bad thing is that Common Lisp has no concept of environment variables. Most Lisps have extensions which allow them to retrieve the values of environment variables. For portability & to isolate this non-Common Lisp code, we'll want to wrap the Lisp-specific function for retrieving an environment variable into a function. For that matter, we might want to hide all the environment variables within a small library with hard-coded parameter names.

Figure 4 shows an abbreviated version of a program that makes a table of the CGI environment variables & their values. The full version of the program is show-env.cgi, & it is also installed on my web server so you can run it there. The `system:getenv` function is specific to clisp; we'll make a portable wrapper shortly.

You might experiment with different arguments. Here are some suggestions:

1. With no arguments, the query string will be empty.
2. With one, non-assignment argument, which isn't exactly a valid URL, I guess.
3. With one pair-wise argument
4. With three pair-wise arguments
5. With three arguments separated by semicolon, which is a non-standard that was proposed but not adopted a while back.

# 8 Forms

An HTML form sends parameters to a CGI program over the same socket as the HTTP request. A form with this article lets you enter many parameters

(all are optional). When you press any of the submission buttons at the bottom of the form, it runs a CGI program which will print verbatim the parameter stream from the form. You can see how the form's parameters are encoded.

# A Is a Web Server Running?

An easy way to check that a web server is running is to connect to it with telnet. If your web server's hostname is *hostname*, then you'd type this on the command line:

```
telnet hostname 80
```

If you get a connection within a few seconds, a web server is running. Type "get index.html". You should see some HTML text in reply. Even if it's an error message that says there is no file called index.html, it's good that you see it. Then the connection should close.

# B Application Delivery

*I've added this section on the spur of the moment. I need to flesh it out or fix it later. – Gene Michael Stover, 11 May 2004*

The techniques I've shown so far have used Lisp as a script language. In other words, they create a unix script file that uses your Lisp or a wrapper (such as my `lisp-script` program) to load & evaluate a Lisp program's source code.

Another way to do it would be to load your CGI program's Lisp sources into your Lisp, then save a "core file".[3] You do this at development time. Then you write a short unix script file that runs your Lisp with that image file & with a single, simple Lisp expression that causes your CGI program to run. That simple unix script might look like this:

```
#! /bin/sh

/my/path/my-lisp -imagefile /my/path/my-cgi.image \
                 -expression "(main)"
```

# C Other File Formats

- This document is available in HTML format http://cybertiggyr.com/lc/.
- This document is available in Pointless Document Format (PDF) at http://cybertiggyr.com/gene/lc/lc.pdf

[3] Core files might also be known as image files.

# References

[1] The common gateway interface. http://hoohoo.ncsa.uiuc.edu/cgi/.

[2] World Wide Web Committee. Cgi: Common gateway interface. http://www.w3.org/CGI/.

[3] D.R.T. Robinson. The www common gateway interface, version 1.1, June 1999. http://cgi-spec.golux.com/draft-coar-cgi-v11-03-clean.html.

[4] Gene Michael Stover. Lisp script. *personal web site*, April 2004. http://cybertiggyr.com/gene/hiil/.