Introduction to LiveCode CGIs - A Tutorial

Debugging Text-based CGIs

Debugging text-based CGIs can be a challenge because very little error information is available. All you see in the browser is a generic "Server Error" message when something goes wrong. Here are some tips for more easily debugging text-based CGIs.

Quick Checklist

Some errors are so common that it is helpful to run through this checklist first before trying other debugging techniques. Check for these common errors:

1. Make sure the first line of the CGI script has the correct path to the engine, and make sure there are no blank lines or other characters before that line. Be sure the `-ui` flags are present.
2. Make sure that the CGI text file, any stacks in use, and the LiveCode engine all have their permissions set to executable. In most cases, this is `chmod 755`.
3. Make sure that line endings in all text files are correct for the server the file is running on.
4. If you are using OS X, make sure Personal Web Sharing is turned on in System Preferences.
5. All data returned by the CGI must be preceded by the correct headers. At a minimum, this must be a "content-type:" declaration. You can add other headers if your script requires it.
6. Make sure that all the correct Apache libraries are installed (see list below.)

Other Debugging Techniques

Write and debug the script in LiveCode as much as possible

Debugging in the LiveCode script editor makes it easier to catch syntax and logic errors. If possible, write as much of the script as you can within the LiveCode IDE and then, after testing, move it to a text file.

Check the server log

The server log is one of your most valuable resources. If any error information is available, it will be listed in the server log. These errors can tell you about missing libraries, the line number where an error occurred, permissions errors, and "file not found" errors, among others. On OS X and the various UNIX platforms, the server log is in `/private/var/log/httpd/`, and you can open it in any text editor. If you are using a remote ISP, ask them how to access your server logs.

In addition to server errors, the server log will also list errors from the LiveCode engine. If the first word of the error message is "revolution" or "livecode" then you know the problem is related to your script, rather than to the server or its setup.

Write scripts in short sections

To debug while interacting with the server, write the CGI script one line or one section at a time, and add temporary lines of code that return data to the browser at various points so you can see what the data is. This also one way to isolate the location in a script where a failure is occuring (although the server log can often tell you the same thing.)

If the script runs without errors but the end results are unexpected, try returning a list of variables and their values at end of the content output so you can see them for reference.

Put the result

Include `put the result` after lines of script that aren't working, or use try/catch structures to identify problems. Any data that is "put" will be displayed in the browser.

Server 500 Errors are usually script errors...

Server error 500, "Internal Server Error", or "Server error: premature end of header" are almost always script errors. If possible, move the script into a stack and test it in the LiveCode IDE for syntax errors. Otherwise use one of the above techniques to locate the problem.

...unless they are library errors

There is one occasion where a Server 500 Error may not be a script error, and that can occur if your server does not have the correct library files installed. In this case, the error in the server log will reference the libraries that are missing, and you'll see something similar to this:

```
Premature end of script headers: /usr/local/apache/cgi-bin/echo.cgi
revolution: error in loading shared libraries: libXext.so.6:
cannot open shared object file: No such file or directory
```

This tells you which library you (or your ISP) will need to install. LiveCode requires these libraries, some of which are often omitted by ISPs:

libXext.so.6
libX11.so.6
libm.so.6
libc.so.6
ld-linux.so.2

Use shell

Run a shell session in a terminal program. First `cd` to the cgi-bin folder and then type `./engineName scriptName`. For example, type `./revolution hello.cgi` to run the "hello.cgi" script.

Determining Environment Variables

Servers are all configured differently, and not all of them support every possible environment variable. The best way to determine which ones your server can work with is to run a CGI script that reports them.

You can download the echo.mt script for this purpose. Unzip this file and place the script into your cgi folder, set its permissions to 755, and call it from your browser like this:

http://localhost/cgi-bin/echo.mt

If you are running the script from a remote server, substitute the IP or domain address of the server for `localhost`.

Other online references

Here are some other sites that discuss using LiveCode as a CGI:

http://forums.runrev.com/phpBB2/viewforum.php?f=15 The RunRev LiveCode forum which discusses LiveCode CGIs.

http://www.navaching.com/pagem.html Nelson Zink discusses setting up a website to do e-commerce using MetaCard or LiveCode as a CGI

http://www.andregarzia.com/index.html Andre Garcia, an expert in LiveCode CGIs, offers additional tools and libraries.

Up to top | Introduction to LiveCode CGIs - TOC |Next Page