CGI Scripting in SWI-Prolog

Under Windows 2003 Server (IIS 6.0)

Michael A. Covington
Artificial Intelligence Center
The University of Georgia
Athens, GA 30602-7415 U.S.A.
www.ai.uga.edu/mc

2007 April 3

Introduction

CGI (Common Gateway Interface) is a feature of most web servers that allows web pages to be generated by running a computer program at the time the page is requested. This is how computation is done on the Web.

In principle, any program can be a CGI script if it can write to standard output, and can read standard input or environment variables if input is needed.

In what follows, I detail one way to set up CGI scripting with SWI-Prolog version 5 (admittedly not the latest version) and Microsoft IIS 6.0 (under Windows 2003 Server).

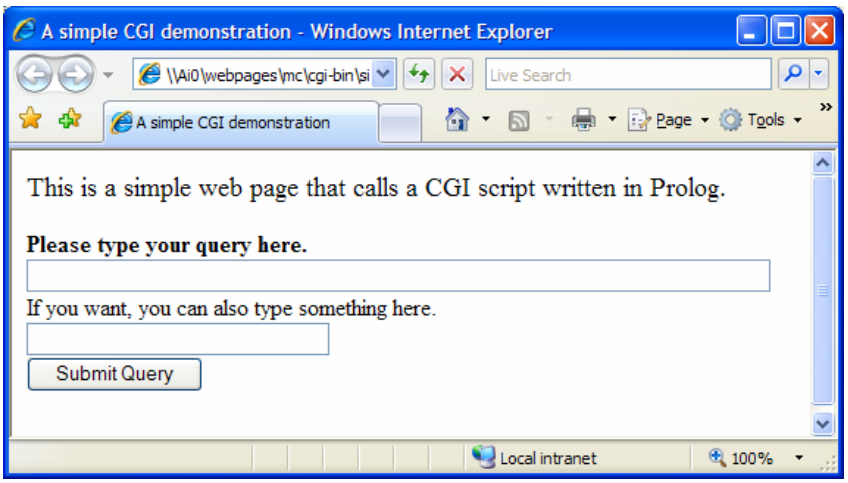
This document is not a complete guide to CGI for the uninitiated; I assume that you will use other sources for general information about CGI and about IIS 6.0.

Example of a web page that calls CGI

Here is an example of the HTML code for a web page that collects data in a form and passes it to a CGI program:

```
<HTML>
<TITLE>A simple CGI demonstration</TITLE>
<BODY>
<big>This is a simple web page that calls a
CGI script written in Prolog.</big>
<FORM METHOD=POST ACTION="simple-example.swi">
<b>Please type your query here.</b>
<br>
<INPUT TYPE="TEXT" NAME="Field1" SIZE=80>
<br>
If you want, you can also type something here.
<br>
<INPUT TYPE="TEXT" NAME="Field2" SIZE=30>
<br>
<INPUT TYPE="SUBMIT">
</FORM>
<P>
</BODY>
```

The program uses the POST method to send input to program *simple-example.swi*, which is given in the next section. When displayed on the screen, the web page looks like this:



When you press *Submit Query*, the program *simple-example.swi* is launched and is given the data in the form fields as input.

Example of a CGI program in Prolog

Here's what *simple-example.swi* looks like:

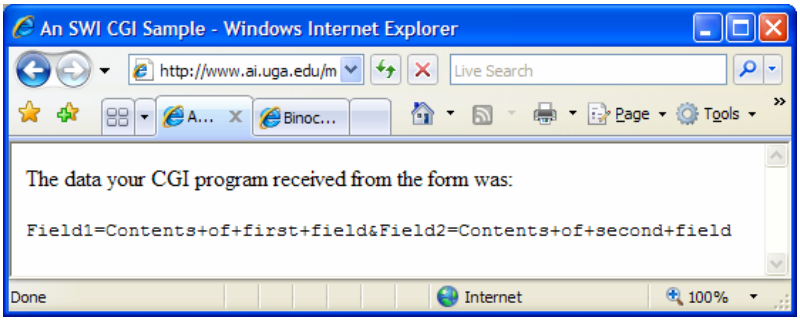
```
% Very simple CGI script in SWI Prolog.
% It receives input from a form
% via the POST method and (in this version)
% just echoes it back to the user.

start :-
  write('Content-type: text/html'),nl,
  nl,
  write('<HTML>'),nl,
  write('<TITLE>An SWI CGI Sample</TITLE>'),nl,
  write('<BODY>'),nl,
  write('The data your CGI program received from the form was:'),nl,
  write('<PRE>'),nl,
  input_post_string(S),     % read the POST data
  output_string(S),
  write('</PRE>'),nl,
  write('</BODY>'),nl,
  write('</HTML>'),nl.


% input_post_string(-String)
%   Accepts line of input as an Edinburgh string (list of ASCII codes).
%   Does not print the '|:' prompt before accepting input.
%   Considerably modified for SWI Prolog under Windows 2000.
%   Uses CONTENT_LENGTH (from the environment) to determine how
%   many characters to read, since SWI Prolog cannot detect end of
%   file on standard input.

input_post_string(String) :-
    getenv('CONTENT_LENGTH',NS),   % NS is an atom like '21' for 21
    atom_to_term(NS,N,_),          % so convert it to number
    !,
    prompt1(''),                   % suppress the '|:' prompt
    input_n_string(N,String).      % and now read N characters

input_post_string([]).
    % if there was no CONTENT_LENGTH value, there's no content

input_n_string(0,[]) :- !.

input_n_string(N,[C|Chars]) :- get0(C), NN is N-1, input_n_string(NN,Chars).


% output_string(+String)
%   Outputs an Edinburgh string to stdout.

output_string([First|Rest]) :-
   put(First),
   output_string(Rest).

output_string([]).
```

To read its input, the program first queries the *CONTENT_LENGTH* environment variable, then accepts exactly that number of characters from standard input without displaying the usual '|:' prompt.

The output of the program is a complete web page, beginning with the *Content-Type* line.

The main predicate is called *start* because that's how we're going to set up the Prolog script engine – we will tell it to pass the query '?- start.' to the freshly loaded Prolog program.

The result of running the program is this:



This is from typing "Contents of first field" and "Contents of second field" in the two fields respectively. Note that blanks come in as plus signs.

Two debugging hints

The Prolog program must not produce any error messages or warnings (e.g., about singleton variables). If it does, they will be output before the *Content-Type* line and IIS will report:

The specified CGI application misbehaved by not returning a complete set of HTTP headers.

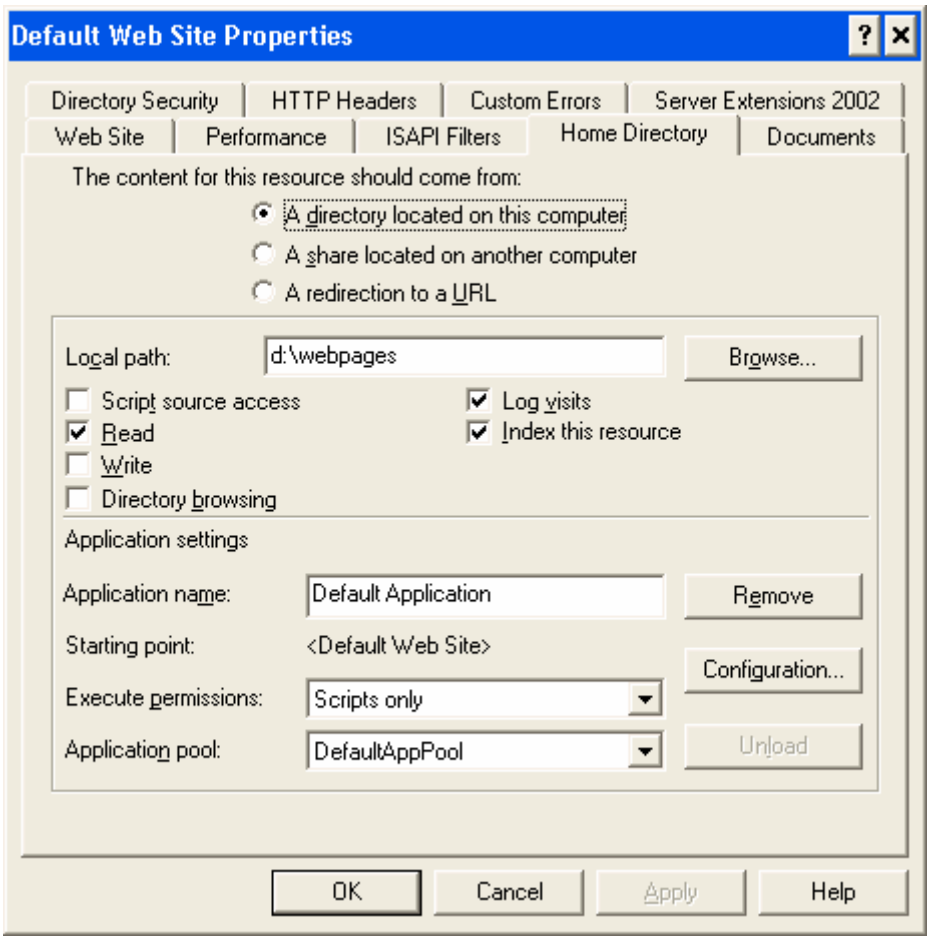
Unfortunately, when this happens, there isn't much you can do about it. Try running your program in a non-CGI environment so that you can see its output. You can set environment variables by using the SET command on the command line.

It is important for the program to output the *Content-Type:* line before it tries to do anything else, so that if there is erroneous output, you'll see it. Sometimes you can troubleshoot a misbehaving program by rigging it to output the *Content-Type:* line before it runs into trouble.

Server configuration

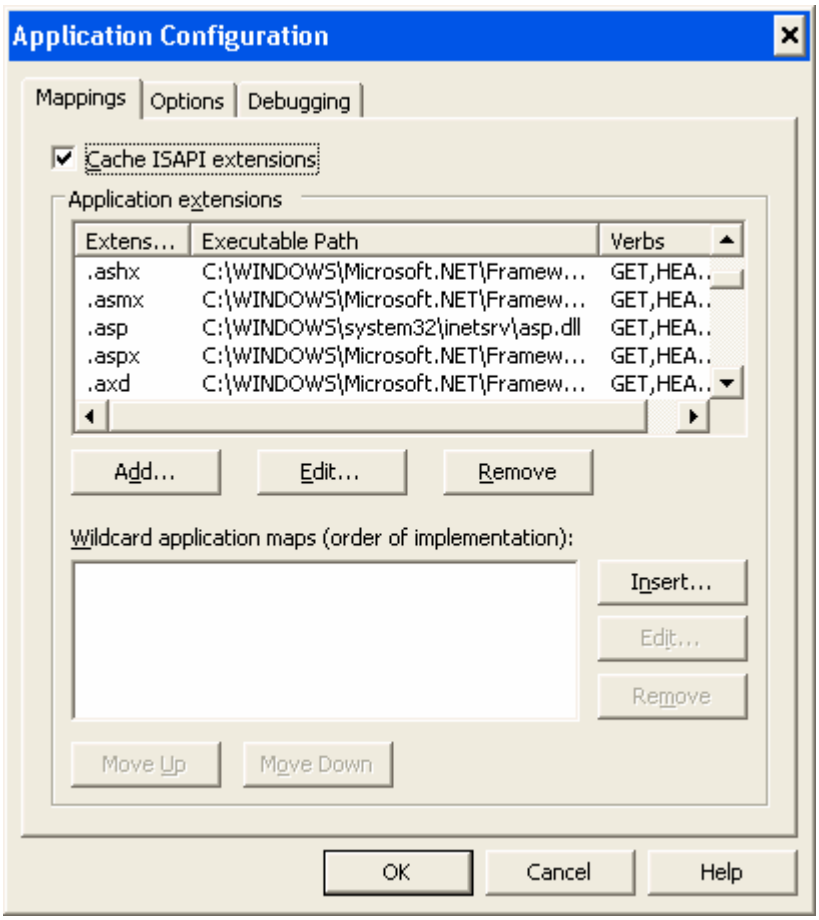
Here is how to tell IIS 6.0 that SWI-Prolog is the "script engine" for executing "scripts" with filenames that end in *.swi* – we didn't use *.pl* because that's for Perl.

On the server, go to Administrative Tools, open Internet Information Services (IIS) Manager, follow the directory tree to Default Web Site, right-click on it, and choose Properties. Bring the Home Directory tab to the front. It should look like this:



The Application Name can be anything you like. If there is none, click Create (in place of Remove).

Then click Configuration. You'll get a window like this:



You're looking at a table of filename extensions and the script engines that handle them. Click *Add* and add an entry with:

Executable:
`"C:\Program Files\pl\bin\plcon.exe" -q -t halt -g start -f %s -- %s`

Extension: .swi

Verbs: GET,POST

Script Engine: Checked

Verify that file exists: Checked

What you are doing is telling IIS how to run an *.swi* file. The "executable" is interpreted as follows (see "command-line options" in the SWI manual):

`"C:\Program Files\pl\bin\plcon.exe"` – The program to launch.

`-q` – Suppress the SWI-Prolog header.

`-t halt` – After the main goal succeeds, execute "?- halt."

`-g start` – As the main goal, execute "?- start."

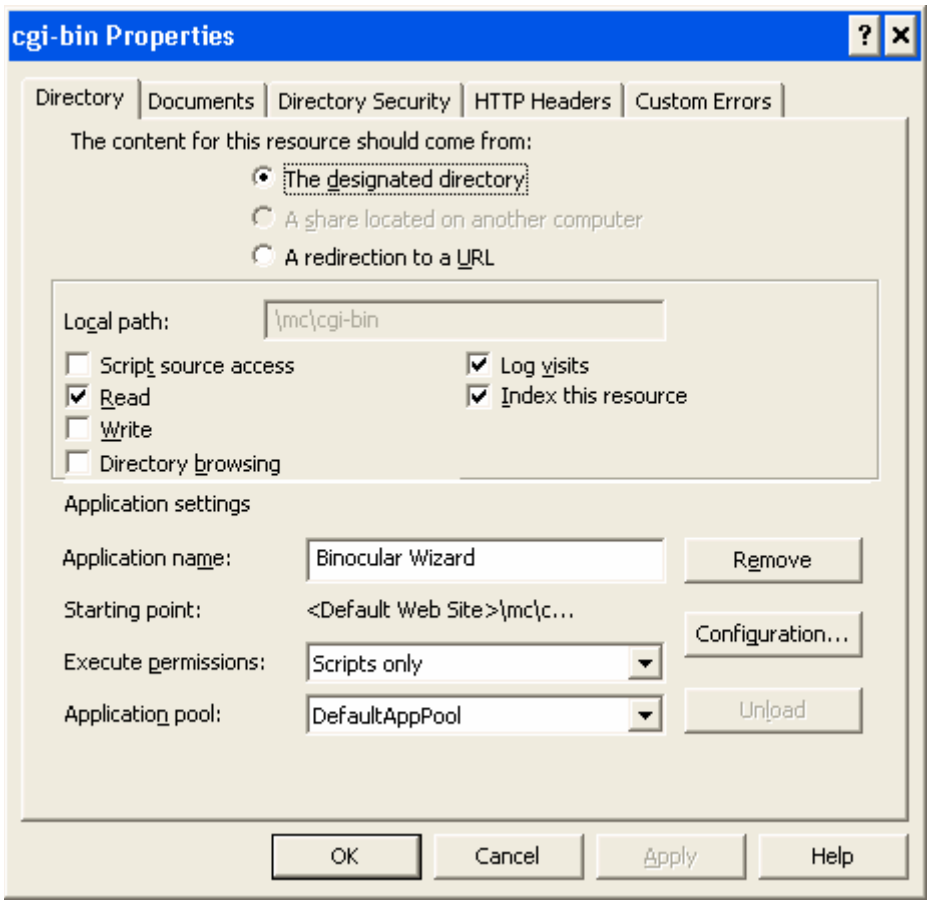
`-f %s` – Start by loading the specified *.swi* file. (%s is a placeholder for where IIS will insert the filename.)

`-- %s` – Also pass the *.swi* filename (and whatever follows it) to the Prolog program as arguments, so that it can parse them if it wants to.

You may want to add more command-line options to enlarge memory or set other parameters.

User directory configuration

You must still enable CGI scripting in the directory where the *.swi* file will reside. To do this, navigate to that directory in IIS Manager, right-click on it, and choose Properties. Then bring the Directory tab to the front (if it isn't already) and make settings like these:



The application name (here “Binocular Wizard”) can be anything you like. By default, there will be none, and you should click *Create* to create one.

If the CGI program creates any files, the account IUSR_*machinename* must have permission to write in the directory where they will be created.
Online demonstrations
CGI scripts written in SWI-Prolog are currently running at:

http://www.ai.uga.edu/mc/wizard.html

http://www.ai.uga.edu/mc/sparse/sparsecgi.html

These are not necessarily permanent.

-end-