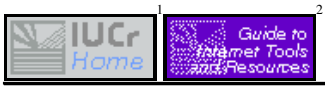

The WWW Common Gateway Interface (CGI)

What is CGI?

The WWW is most commonly used for the presentation of static information. Users use their browser to look at the information provided on the HTML pages of interest. More and more often, however, the type of information one obtains is dynamic. This means that the information in the format that you view was only generated at the time it was requested. A common example of this is the many search engines available on the WWW. For these searches, the sequence of events is:

1. the user enters keywords,
2. the keywords are returned to the server,
3. the server uses the keywords to run a search through its internal database,
4. the server uses the search result to write a custom HTML document and
5. the custom HTML document is returned to the user for display in a browser.

The *dynamic information* displayed is generated as a result of the request. Another example can be seen in the online WWW registration forms most of us have encountered with recent conferences, such as the *Small Molecules Indaba II*[3]. Here users enter their registration information on the WWW and this information will then be provided to the conference organizers in a systematic way.

CGI can then be characterized by a few key features:

- Information is dynamic – a program is actually run every time the information is requested.
- Processing is Server Based – all additional calculations and database searches performed happen on the server where the web site is. This is in contrast to client side programs like Java[4] and VRML[5] where the user's computer does a considerable amount of extra work.

How CGI works

The easiest way to understand how CGI works is to follow a step–by–step example. Let us consider the case of a WWW search of the World Directory of Crystallographers[6].

1. The user will access a document containing a WWW form. This allows the user to enter information directly into the WWW browser[7] interface. The form will contain a button to start the search, an example being the general search form provided by the IUCr for the WDC9[8]. In this form, four fields are available for entering data. The first of these is the *family name* field which is a good choice to try.
2. When the user enters the family name into the form, the browser associates the entry with a variable name defined in the form. If you entered the text *flack* into the family name field it will have been associated with the variable *Qfamily_name*.
3. When the user clicks on the *Submit Query* button (normally placed towards the bottom of the form), this information is returned to the server in a standard way known as the common gateway interface. This standard allows a fair amount of information to be passed to a program on the server computer.
4. When the server receives the information from the client, it executes the program requested by the client. In the case of the WDC9, this program is the C program *wdcquery*.

[1] http://www.iucr.ac.uk/iucr–top/welcome.html
[2] int.index.html
[3] http://www.gh.wits.ac.za/indaba
[4] int.java.html
[5] int.vrml.html
[6] int.wdc.gen.html
[7] int.browsers.html
[8] http://www.iucr.ac.uk/cgi–bin/wdcquery

The information entered by the user is made available to this program in the form of environmental variables. There are over a dozen of these commonly used by CGI programs, so we'll look only at those relevant to the current example.:
Variabledescriptionexample
REQUEST_METHODType of form requestGET
QUERY_STRINGUser data for GET method
Jserver=www.iucr.ac.za&Qfamily_name=flack
CONTENT_LENGTHLength of data 41
Much of the rest of the information passed will either be blank, or concern information like browser type and version.

5. It is entirely up to the CGI program, in this case *wdcquery*, to extract and interpret this information from the environment variables and act accordingly. *Wdcquery* will then use this information to access the server described by the form variable *Jserver* using the network database system *phf* by Jim Browne. This will return the results of the WDC9 database search to *wdcquery*, which will then reformat the answer in HTML and write it to the standard output.
6. The server takes all information written to the standard output by the CGI program and returns it directly to the client. This means that all search results that *wdcquery* reformatted in HTML will be returned to the original browser for viewing.

If all this happens fast enough, it will appear to the client just as if they accessed a simple HTML page of information.

One important comment about the above procedure relates to the way that the *wdcquery* program accessed the WDC9 database. This was done through a network database access completely unrelated to the WWW. This example, therefore, emphasizes the fact that the CGI program can be a Gateway to any external system. The University of Western Washington in Bellingham, for example, have their WWW server connected to a video digitizing facility, allowing users from around the world to take pictures of their fountain square with the *click–of–a–button ...*

A Simple Example using the POST method

This simple form will allow you to enter your name and email address into two fields which will then be stored in a simple database on the WWW server.

Firstly, let's look at the HTML code for the form:

```
<FORM ACTION="/cgi-bin/xtras/email-data" METHOD="POST">
<INPUT TYPE="text" NAME="name" SIZE="50">Enter your name<BR>
<INPUT TYPE="text" NAME="email" SIZE="50">Enter your email address<BR>
<INPUT TYPE="submit" VALUE="Add to Database">
<INPUT TYPE="reset" VALUE="Clear data from form">
</FORM>
```

Now, here is the form as displayed by your browser (*the table border was added for clarity*):

Enter your name

Enter your email address

In the POST method the data you enter is not provided to the program in the environment variable QUERY_STRING but is simply read from the programs standard input. This allows for a more streamlined data flow from the server to the program, without the variables appearing in the URL as they do for the GET method.

[Index][9] – 11th September 1996 – © B. Craig Taverner[10] **– Not to be copied or reproduced without permission**

[9] int.index.html
[10] http://www.gh.wits.ac.za/craig/