Passing Parameters via CGI

To pass parameters via CGI, you need a form (html page). This feature is convenient because it lets programs be called via simple links, not just by full-blown forms. To test this out, take the original URL and add a question mark followed by the parameter name, an equal sign, and the value desired. For example, the following URL would call the *hello* script with the `name` parameter set to the value `Ashok Kumar`:

```
http://www.ashok.com/cgi-bin/hello.pl?name=Ashok+Kumar
```

When you point your browser at this URL, the browser not only requests the web server to invoke the `hello.pl` program, but it also passes the string `name=Ashok+Kumar` to the program. Now it's up to the program to read the argument string and pick it apart.

Enter the *CGI.pm* module, which always parses the incoming CGI request correctly. To pull this module into your program, merely say:

```
use CGI;
```

somewhere near the top of your program.

The `use` statement is somewhat like a `#include` statement in C programming in that it pulls in code from another file at compile time. But it also allows optional arguments specifying which functions and variables you'd like to access from that module. Put those in a list following the module name in the `use` statement. You can then access the named functions and variables as if they were your own.

In this case, all we need to use from *CGI.pm* is the `param()` function.

If given no arguments, `param()` returns a list of all the fields that were in the HTML form that this CGI script is responding to. (In the current example, this list contains the `name` field. In general, the list contains all the names in `name=value` strings received from the submitted form.) If given an argument naming a field, `param()` returns the value (or values) associated with that field. Therefore, `param("name")` returns "`Ashok Kumar`", because we passed in `?flavor=Ashok+Kumar` at the end of the URL.

Even though we have only one item in our import list for use, we'll employ the `qw()` notation; this way it will be easier to expand the list later:

```
#hello.pl: program to welcome user
use CGI qw(param);
print <<END_of_Start;
Content-type: text/html

<HTML>
 <HEAD>
  <TITLE>Hello User</TITLE>
 </HEAD>
 <BODY>
  <H1>Greetings, Terrans!</H1>
END_of_Start

my $name = param("name");
print "<P>Welcome $name!";
print <<All_Done;
 </BODY>
</HTML>
```

```
All_Done
EOF
```

Form Generation:

Perhaps you're tired of typing your program's parameter to your browser. Just make a fill-out form instead, which is what most folks are used to. The parts of the form that accept user input are typically called *widgets*, a much handier term than graphical input devices. Form widgets include single- and multiline textfields, pop-up menus, scrolling lists, and various kinds of buttons and checkboxes.

Create the following HTML page, which includes a form with one textfield widget and a submit button. When the user clicks on the submit button, the *hello* script specified in the `ACTION` tag will be called:

```
<!-- hello.html -->
<HTML>
 <HEAD>
  <TITLE>Hello User</TITLE>
 </HEAD>
 <BODY>
  <H1>Hello User</H1>
  <FORM ACTION="http://www.biogem.org/cgi-bin/hello.pl">
  What's your name? <INPUT NAME="name" TYPE="text">
  <INPUT TYPE="submit">
  </FORM>
 </BODY>
</HTML>
```

Remember that a CGI program can generate any HTML output that you want, which will then be passed to any browser that fetches the program's URL. A CGI program can, therefore, produce the HTML page with the form on it, just as a CGI program can respond to the user's form input. Moreover, the same program can perform both tasks, one after the other. All you need to do is divide the program into two parts, which do different things depending on whether or not the program was invoked with arguments. If no arguments were received, then the program sends the empty form to the browser; otherwise, the arguments contain a user's input to the previously sent form, and the program returns a response to the browser based on that input.

Keeping everything in a single CGI file this way eases maintenance. The cost is a little more processing time when loading the original page. Here's how it works:

```
#!C:\Perl\bin\perl.exe
use CGI qw(:standard);
my $name = param("name");
print header, start_html("Hello User"), h1("Hello User");
if ($name)
{
 print p("Welcome $name!");
}
else
{
 print hr(), start_form();
 print p("What's your name? ", textfield("name"));
 print p(submit("Sumit"), reset("Reset"));
 print end_form(), hr();
}
```

Now, fill in the flavor field and press Submit button.