CGI Programming

What is "CGI"?

- Common Gateway Interface
- A means of running an executable program via the Web.
- CGI is not a Perl-specific concept. Almost any language can produce CGI programs
  - even C++ (gasp!!)
- However, Perl does have a *very* nice interface to creating CGI methods

How Does it Work?

- A program is created, like normal.
- The program must be made user-executable
- A user visits a web page.
- The web browser contacts the server where the CGI program resides, and asks it to run the program
  - passes Parameters to the program
    - similar to command line arguments
- The server runs the program with the parameters passed in
- The server takes the output of the program and returns it to the web browser.
- The web browser displays the output as an HTML page

Forms

- Most (not all) CGI scripts are contacted through the use of HTML forms.
- A form is an area of a web page in which the user can enter data, and have that data submitted to another page.
- When user hits submit button on the form, web browser contacts the script specified in the form tag.

Creating a Form

```
<form method="post" action="file.cgi">
…
<input type="submit" value="Submit Form">
</form>
```

- Method attribute specifies how parameters are passed to the CGI program. "post" means they're passed in the HTTP header (and therefore aren't seen anywhere). "get" means they're passed through the address bar in the web browser.
- Action specifies the program you want the web browser to contact.
- <input> is tag used to accept User data. type="submit" specifies a Submit button. When user clicks this button, browser contacts file specified in action attribute.

Form Input Types

- Many different ways of getting data from user. Most specified by <input> tag, type specified by `type` attribute of <input>
- text ➔ a text box
- checkbox ➔ a check box
- radio ➔ a Radio button
- password ➔ password field
  - (text box, characters display as ******)
- hidden ➔ hidden field (nothing displayed in browser)
- submit ➔ Submit button. Submits the form
- reset ➔ Reset button. Clears form of all data.
- button ➔ A button the user can press
  - (usually used w/ javaScript. *shudder*)
- file ➔ field to upload a file
- image ➔ an image user can click to submit form

Other Attributes of <input>

- name ➔ name of input field.
- value ➔ value returned from checks & radios, text of submits and buttons, contents of text, password, and hidden
- size ➔ width of text or password
- checked ➔ radio or checkbox 'turned on'
- src ➔ URL of an image

Inputs that Don't Use <input>

- <textarea> - big text field. Can specify rows and cols attributes
- <select> - create a drop-down menu.
  - <option value="…"> Options in the drop down menu.

Great. We can input. Now what?

- Now, we can write a CGI program that takes those inputs and *does stuff* with them.
- This is still a perl script. Therefore, still need the shebang as top line of the code.
- Next, need to include all the CGI methods. These are stored in CGI.pm
  - As you may guess, TMTOWTDI.
- `use CGI;`
- `use CGI ":standard";`

What's the difference?

- Function oriented vs. Object Oriented
- CGI.pm actually defines a class. Therefore, if you `use CGI;` you can then do
- `$query = new CGI;`
- and access the CGI subroutines as methods of $query.
- Alternatively, you can get a 'standard' set of CGI functions to call directly
- `use CGI ":standard";`
- Now can call CGI functions without fiddling with objects
- All my examples will be function oriented. (for now)

Outputting from CGI

- Just as CGI program took 'input' from user via web browser, it outputs back to use via web browser as well.
- STDOUT is redirected to the web browser that contacted it.
- This means you don't have to learn any new output functions. print() will now throw data to the web browser.

Beginning a CGI Program

```
#!/usr/local/bin/perl
use CGI ":standard";
print header("text/html");
```

- header() prints HTTP header to web browser. Argument is MIME type of data. Defaults to text/html, so you can usually just leave the argument out.

Now Create your Output

- Remember, you're building an HTML page in the output. So you must follow the HTML format:

```
print "<html><head>",
"<title>My CGI Program</title>\n",
"</head><body>\n";
```

- CGI.pm gives you a better way to do this. We'll get to it soon.

Where'd Those Inputs Go?

- They were passed into the CGI program as parameters. You can retrieve them using the param() function.
- Called in list context w/ no argument, returns names of all parameters.
- Called in scalar context, takes name of one parameter, and returns value of that parameter
- Called in list context w/ an argument, returns array of all values for that parameter (ie, for checks and radios)

dump()

- subroutine defined in CGI.pm. Retrieves list of parameters and creates an HTML list of all parameters and all values.
- Like most CGI functions, doesn't print anything. You must manually print the return value of the function call.
- `print dump;`

CGI.pm → HTML Shortcuts

- CGI gives you methods to create HTML code without actually writing HTML.
- most HTML tags aliased to CGI functions.
- unpaired tags
  - `print br(); # sends <br> to browser`
  - `print p;  # sends <p>  to browser`
- paired tags
  - `print b("Bold text"); #<b>Bold text</b>`
  - `print i("Italic"); #<i>Italic</i>`

More shortcuts

- tags with attributes: place name/value attributes in hash reference as first argument. String enclosed in tag is second argument. Ex:

```
a({href=>"sample.html", target=>"top"},
 "Sample file");
```

- Produces:
- `<a href="sample.html" target="top">Sample file</a>`

start_html()

- Takes one parameter, the title.
- `print start_html("My title");`
- Produces:
- `<html><head><title>My title</title></head><body>`
- `print end_html;`
- Produces:
- `</body></html>`

HTML Form Shortcuts

- For full list of Form shortcuts, see http://stein.cshl.org/WWW/software/CGI/#forms
- Each one takes parameters as name/value pairs. Name starts with a dash. Most parameters are optional
- startform(-method=>“post”, -action=>“foo.cgi”)
  - default method is post. default action is current script
    - this is *very* beneficial.
- produces: <form method=“post” action=“foo.cgi”>
- endform(); # produces </form>

Input Shortcuts

- `textfield(-name=>“MyText”, -default =>“This is a text box”)`
- produces:
- `<input type=“text” name=“MyText” value=“This is a text box”>`
- All HTML Form input shortcuts are similar. Again, see http://stein.cshl.org/WWW/software/CGI/#forms for full list and description.

Programmer Beware

- “default” in input methods is value used *first* time script is loaded only. After that, they hold the values they had the last time the script was run.
- to override (ie, force default value to appear), set parameter `-override=>1` inside input method:
- `textfield(-name=>“foo”, -default =>“bar”, -override=>1);`

Avoid Conflicts

- Some HTML tags have same name as internal Perl functions. Perl will get very confused if you try to use the CGI shortcut methods to print these tags…
- <tr> – table row. conflicts with tr///
  - use Tr() or TR() instead
- <select> – dropdown list. conflicts with select().
  - use Select() instead
- <param> - pass parameters to Java applet – conflicts with param().
  - use PARAM() instead
- <sub> - Make text subscripted. Conflicts with sub keyword.
  - Use Sub() instead

Running CGI on CS machines

- There are two different CGI servers on the CS system. cgi.cs.rpi.edu and cgi2.cs.rpi.edu
- I've been told by labstaff that cgi.cs is depreciated and I shouldn't use it.
  - I'm ignoring them.
- To run on cgi.cs:
  - make file user-executable
  - put in public.html directory
  - go to http://cgi.cs.rpi.edu/~yourID/yourscript.cgi
- To run on cgi2.cs
  - make file user-executable
  - put in public.html/cgi-bin directory
  - go to http://cgi2.cs.rpi.edu/~yourID/cgi-bin/yourscript.cgi
- I find cgi.cs to be more user friendly. cgi2.cs gives me many unexplained errors.

Debugging

- Debugging a CGI program can be one of the most frustrating tasks ever.
- If there are any errors whatsoever, the web browser will simply display `500 Internal Server Error` with no helpful information.
- Solution is to run the program from the command line.
- Perl will ask you for name=value pairs of parameters. Enter each name and value of the parameters, separated by newline. Then press CTRL-D.
- This way, you get the compiler errors, and you can see the 'pure' HTML output of your CGI script.

Lastly…

- Well, lastly for today anyway.
- One common method of CGI programming is to make both the form and the response all in one script. Here's how you do that…

```
#!/usr/local/bin/perl
use CGI ":standard";
print header;
if (!param()){
  print start_html(-title => "Here's a form");
  print startform; #no action
  #create your form here…
} else {
  print start_html(-title=> "Here's the result");
  #display the results of the submitting form
}
```