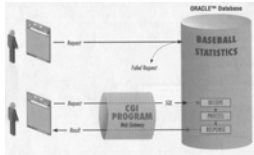


Common Gateway Interface (CGI)

- Access to dynamic server-side content
 - Services
 - Data



CGI Model (Pieces)

- Clients: web browser
- Server: web server mediates communication between browsers and handler programs
- Handler programs: any executable residing on the web server
 - CGI *Protocol*: Specifies interaction between
 - Browser and handler programs
 - Function call syntax
 - Web server and handler programs

CGI Example

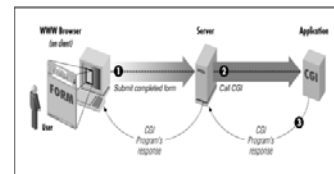
<http://localhost/cgi-bin/hello.pl>

This causes the execution of the Perl script hello.pl

```
#!/local/bin/perl
print "Content-type: text/html\n\n";
print "<html>\n";
print "<body>\n";
print "<h1>hello</h1>\n";
print "</body>\n";
print "</html>\n";
```

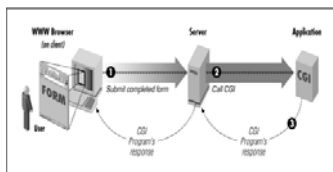
Note: Although our examples use Perl, CGI scripts can be written in any language: C, C++, VB, Python, SmallTalk, Assembly, Lisp etc...

CGI Interaction



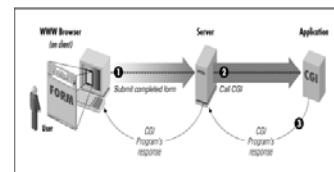
- Client sends request by specifying a URL+additional info.
- Web server receives the request.
- Web server identifies the request as a CGI request
- Web server locates the program to run

CGI Interaction



- Web server starts up the handling program (heavy weight process creation!!)
- Web server feeds request parameters to handler (through stdin or environment variables).

CGI Interaction



- Handler executes
- Output of the handler is sent via stdout back to the webserver for rerouting back to the requesting web browser.
- Output is typically a web page.
- Handler terminates.

Client-Handler Interaction

- Clients request the execution of a handler program by means of a:
 - A request method (e.g., GET, POST)
 - Universal Resource Identifier (URI)
 - Identifies handler
 - Extra arguments

URI Syntax

- Defined in section 2.1 of RFC 1808

<protocol>://<host><port>/<path-info><script>"?"<query-string>

http://finance.yahoo.com/q?a=1&b=2

<protocol>	http
<host>	finance.yahoo.com
<port>	80
<path-info>	null
<script>	q
<query-string>	a=1, b=2

Query String Encoding

- RFC 2396
- Why encode?
 - Can think of a CGI script as a function, send arguments by specifying name/value pairs.
 - Way to pack and unpack multiple arguments into a single string

Encoding Rules

- **All arguments** will be concatenated into a single string of ampersand (&) separated name=value pairs, one pair for each form tag. Like this:
name_1=value_1&name_2=value_2&...
- **Spaces** in a name or value are replaced by a plus (+) sign. This is because url's cannot have spaces in them and under **METHOD=GET**, the form data is supplied in the query string in the url.
- **Other characters** (ie, =, &, +) are replaced by a percent sign (%) followed by the two-digit hexadecimal equivalent of the punctuation character in the ASCII character set.
 - Otherwise, it would be hard to distinguish these characters inside a variable from those between the variables in the first rule above.

Method

- GET
 - Arguments appear in the URL after the ?
 - Can be expressed as a URL
 - Limited amount of information can be passed this way
 - URL may have a length restriction on the server
 - Arguments appear on server log
- POST
 - Arguments are sent in the HTTP message body
 - Cannot be expressed as URL
 - Arbitrarily long form data can be communicated (some browsers may have limits (i.e. 7KB)).
 - Arguments usually does not appear in server logs.

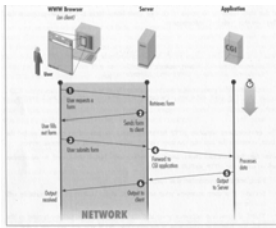
Example

- In Web page

```
<a href="cgi-bin/hello.pl?a=1&n=2"> Click to run CGI </a>
```
- Request sent to web server

```
get /cgi-bin/hello.pl?a=1&b=2 http/1.0
```

Forms and CGI



- Specify request method in “method” attribute
- Automatically encodes all field values

Example

- In Web page

```
<form action="cgi-bin/login.pl" method="get">
  <input type="text" name="idxyz" />
  <input type="text" name="password">
</form>
```
- Request sent to web server
 - Assuming user types 'myID' and 'secret' in fields

```
get /cgi-bin/login.pl?idxyz=myID&password=secret http/1.0
```

CGI Web Server-Handler Interaction

- Information about a request comes from
 - the request header
 - associated message-body (POST)
- Handler Input
 - Meta-variables
 - Stdin (message body)
- Handler Output
 - Stdout

Meta-variables

AUTH_TYPE	REMOTE_HOST
CONTENT_LENGTH	REMOTE_IDENT
CONTENT_TYPE	REMOTE_USER
GATEWAY_INTERFACE	REQUEST_METHOD
PATH_INFO	SCRIPT_NAME
PATH_TRANSLATED	SERVER_NAME
QUERY_STRING	SERVER_PORT
REMOTE_ADDR	SERVER_PROTOCOL
	SERVER_SOFTWARE

Example – Print Environment Variables & Stdin

```
#!/local/bin/perl

print "Content-type: text/html\n\n";
print "<pre style=font: bolder 24pt>\n\n";
print "Environment\n";
@keys = keys %ENV;          # variable names
@values = values %ENV;      # variable values
while (@keys) {
    print pop(@keys), '=', pop(@values), "\n";
}

print "STDIN\n";

while(<>){
    print;                    # request body
}
```

Example – Print Environment Variables & Stdin

```
<body>
<h3>GET</h3>
<form action="cgi-bin/stdin.pl" method="get">
  Student Number: <input type="text" name="studentNumber"> <br />
  Id: <input type="text" name="id"> <br />
  Password: <input type="password" name="password"> <br />
  Return Files: <input type="checkbox" name="returnFiles"> <br />
  <input type="submit" name=".submit" />
</form>

<h3>POST</h3>
<form action="cgi-bin/stdin.pl" method="post">
  Student Number: <input type="text" name="studentNumber"> <br />
  Id: <input type="text" name="id"> <br />
  Password: <input type="password" name="password"> <br />
  Return Files: <input type="checkbox" name="returnFiles"> <br />
  <input type="submit" name=".submit" />
</form>
</body>
```

Example – Environment Variables

Request
get /cgi-bin/environment.pl?var1=val1&var2=val2 http/1.0

Reply
HTTP/1.1 200 OK
Date: Thu, 06 Sep 2001 03:10:04 GMT
Server: Apache/1.3.19 (Win32)
Connection: close
Content-Type: text/html

```
<pre>
Environment
SERVER_PORT=80
REQUEST_METHOD=get
.....
QUERY_STRING=var1=val1&var2=val2&var3=val3
STDIN
h</pre>
```

chocolate.pl

```
#!/usr/local/bin/perl
use CGI;
my $cgi = new CGI;

print "Content-Type: text/html\n\n";
print "<html><body>";

print "This is your order:";

my @params = $cgi->param();

print "<TABLE border='1' cellspacing='0' cellpadding='0'>". "\n";

foreach my $parameter (@params) {
    print "<tr><th>$parameter</th><td>" .
        $cgi->param($parameter) .
        "</td></tr>\n";
}

print "</table>";
print "Your total is $10 gazillion <br/>";
print "Enjoy your movie!";
print "</body></html>";
```

Apache CGI

- Apache directories
 - cgi-bin .cgi files
 - htdocs .html, .gif, .jpeg, .css, .js
 - conf configuration files
 - httpd.conf
 - Main Apache configuration file
 - CGI are disabled by default
 - To enable CGI:
 - Add "ExecCGI" to the "Options" directive
- Options Indexes FollowSymLinks [ExecCGI](#)

topScore.pl

```
#!/local/bin/perl
use CGI qw(:standard);

$name = param('name');
$score = param('score');

#replace white spaces
$name =~ tr/\s/ /;

$entry = $name . " " . $score;

#add new record to file
"echo $entry >> myfile";

#sort records based on highest score
@records = `sort -r -n -k 2,2 myfile`;

print header;
start_html("Top Scores");
print h1("Top Scores");
print "<table>";

foreach $i (0..$#records) {
    if ($records[$i] =~ m/(.+)(\d+)/) {
        print "<tr><td>$1</td><td>$2</td></tr>";
    }
}

print "</table>";
print end_html();
```