Desarrollo de Aplicaciones en Red

José Rafael Rojano Cáceres
http://www.uv.mx/rrojano

CGI

CGI (1)

- The Common Gateway Interface (CGI) represent a *interface* between the Server and the HTTP protocol.
- It let us:
  - *Given a URL invoke a program on the server*
  - *Pass information from HTTP protocol to the program*
  - *Send information from a program to the client*
- Some of the most used languages for CGI are Perl, C, and even shell.

CGI (2)

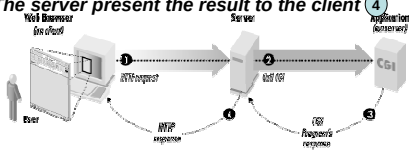
- *Each time a request is made to a CGI the server forks and exec an instance for CGI program*
- *A CGI get information from environment variables, the most important are:*
  - REQUEST_METHOD
  - QUERY_STRING
  - CONTENT_LENGTH

Making a CGI request

- *A Get request can be made for* (1)

```
GET /cgi/hola.cgi HTTP/1.1
```

- *An fork and exec process is throw* (2) *and server pass some parameters*
- *The application send the result that the Web Server collect* (3)
- *The server present the result to the client* (4)



Input and Output

- *CGI get input from STDIN and environment variables*
- *CGI return data to STDOUT*
  - An special line (header) on CGI indicating the return for Web Server
    - Content-type: text/html

STDIN

- *When the Web Server receives a request for a CGI it removes the header an send the body to the script*
  - Be careful to read data from GET, it would be empty making hang the script
  - For POST request always should be read Content-Length for the number of bytes

7

STDOUT

- *As we said a CGI send results to the Web Server through stdout, the server will wait until all the information has been send to it, later it would send to the client.*
- *CGI does not need to write full headers just the header Content-type*

8

STDOUT

Examples

9

CGI in bash

```
#!/bin/sh
echo Content-type: text/html
echo
echo '<html><head><title> hola.cgi </title></head>
echo '<body><h1>Hola mundo!</h1></body></html>'
```

10

CGI in Perl

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html>";
print "<head> <title> hola.pl </title> <head> \n";
print "<body><h1>Hola mundo!</h1> </body>";
print "</html> \n";
```

11

CGI in C

```
#include <stdio.h>
main (int argc, char **argv)
{
      printf ("Content-type: text/html\n\n");
      printf ("<html>");
      printf ("<head> <title> hola.c </title> <head> \n");
      printf ("<body><h1>Hola mundo!</h1> </body>");
      printf ("</html> \n");
}
```

12

STDIN in Perl

- *The way in which CGI receives information from Web Server is through environment variables.*
- *These are got by:*

  %ENV

Getting some information

```
#!/usr/bin/perl -wT
print << END_OF_HTML;
Content-type: text/html

#!/usr/bin/perl -wT
print <<END_OF_HTML;
Content-type: text/html
<HTML>
<HEAD>    <TITLE>About this Server</TITLE> </HEAD>
<BODY>
<H1>About this Server</H1>
<HR>
<PRE>
  Server Name:       $ENV{SERVER_NAME}
  Listening on Port: $ENV{SERVER_PORT}
  Server Software:   $ENV{SERVER_SOFTWARE}
  Server Protocol:   $ENV{SERVER_PROTOCOL}
  CGI Version:       $ENV{GATEWAY_INTERFACE}
</PRE>
<HR>
</BODY>
</HTML>
END_OF_HTML
```


Getting some information

```
#!/usr/bin/perl -wT
print << END_OF_HTML;
Content-type: text/html
<HTML>
<HEAD>    <TITLE>About this Server</TITLE> </HEAD>
<BODY>
<H1>About this Server</H1>
<HR>
<PRE>
  Server Name:       $ENV{SERVER_NAME}
  Listening on Port: $ENV{SERVER_PORT}
  Server Software:   $ENV{SERVER_SOFTWARE}
  Server Protocol:   $ENV{SERVER_PROTOCOL}
  CGI Version:       $ENV{GATEWAY_INTERFACE}
</PRE>
<HR>
</BODY>
</HTML>
END_OF_HTML
```


STDIN in Shell

- *In the Shell STDIN is gotten by the environment variables using the special symbol $*
- *For example:*

  $QUERY_STRING

Getting some information

```
#!/bin/sh
echo Content-type: text/html
echo
echo "<HTML>"
echo "<HEAD>    <TITLE>About this Server</TITLE> </HEAD>"
echo "<BODY>"
echo "<H1>About this Server</H1>"
echo "<HR>"
echo "<PRE>"
echo Server Name:       $SERVER_NAME
echo Listening on Port: $SERVER_PORT
echo Server Software:   $SERVER_SOFTWARE
echo Server Protocol:   $SERVER_PROTOCOL
echo CGI Version:       $GATEWAY_INTERFACE
echo "</PRE>"
echo "<HR>"
echo "</BODY>"
echo "</HTML>"
```


STDIN in C

- *In language C variables are got by getenv*
- *And the parameter is the name of the environment variable, example:*
  - Getenv ("QUERY_STRING");
- *char *getenv(const char *nombre);*
- *http://www.conclase.net/c/librerias/function.php?fun=getenv*

Getting some information

```
#include<stdio.h>

int main(int argc, char **argv)
{
  printf("Content-type: text/html\n\n");
 printf("<HTML>");
 printf("<HEAD> <TITLE>About this Server</TITLE> </HEAD>");
 printf("<BODY>");
 printf("<H1>About this Server</H1>");
 printf("<HR>");
 printf("<PRE>");
 printf("  Server Name:%s",       getenv("SERVER_NAME"));
 printf("  Listening on Port:%s", getenv("SERVER_PORT"));
 printf("  Server Software:%s",   getenv("SERVER_SOFTWARE"));
 printf("  Server Protocol:%s",   getenv("SERVER_PROTOCOL"));
 printf("  CGI Version:%s",       getenv("GATEWAY_INTERFACE"));
 printf("</PRE>");
 printf("<HR>");
 printf("</BODY>");
 printf("</HTML>");
 return 0;
}
```


Environment Variables

Environment Variable	Description
AUTH_TYPE	The authentication method used to validate a user. This is blank if the request did not require authentication.
CONTENT_LENGTH	The length of the data (in bytes) passed to the CGI program via standard input.
CONTENT_TYPE	The media type of the request body, such as "*application/x-www-form-urlencoded* ".
DOCUMENT_ROOT	The directory from which static documents are served.
GATEWAY_INTERFACE	The revision of the Common Gateway Interface that the server uses.
PATH_INFO	Extra path information passed to a CGI program.
PATH_TRANSLATED	The translated version of the path given by the variable PATH_INFO.
QUERY_STRING	The query information from requested URL (i.e., the data following "?").
REMOTE_ADDR	The remote IP address of the client making the request; this could be the address of an HTTP proxy between the server and the user.
REMOTE_HOST	The remote hostname of the client making the request; this could also be the name of an HTTP proxy between the server and the user.
REMOTE_IDENT	The user making the request, as reported by their ident daemon. Only some Unix and IRC users are likely to have this running.
REMOTE_USER	The user's login, authenticated by the web server.
REQUEST_METHOD	The HTTP request method used for this request.
SCRIPT_NAME	The URL path (e.g., */cgi/program.cgi*) of the script being executed.
SERVER_NAME	The server's hostname or IP address.
SERVER_PORT	The port number of the host on which the server is listening.
SERVER_PROTOCOL	The name and revision of the request protocol, e.g., "HTTP/1.1".
SERVER_SOFTWARE	The name and version of the server software that is answering the client request.

Server configuration

- ***Enabling CGI execution with Apache is very simple, place this directive in httpd.conf :***

  ScriptAlias /cgi-bin /usr/local/apache/cgi-bin

- ***The directory that holds CGI scripts must be outside the server's document root***

  <Directory " /usr/local/apache/cgi-bin">
  AllowOverride None
  Options None
  Order allow,deny
  Allow from all
  </Directory>

If you want for one user

AddHandler cgi-script .cgi .pl

ScriptAliasMatch /cgi-bin/ "/home/user/public_html/cgi-bin"

<Directory /home/*/public_html/cgi-bin>
AllowOverride None
Options +ExecCGI +Includes +Indexes
Order allow,deny
Allow from all
</Directory>

If you want for * user

AddHandler cgi-script .cgi .pl

ScriptAliasMatch ^/~([^/]*)/cgi-bin/(.*) /home/$1/public_html/cgi-bin/$2

<Directory /home/*/public_html/cgi-bin>
AllowOverride None
Options +ExecCGI +Includes +Indexes
Order allow,deny
Allow from all
</Directory>

Reference

- ***Web 2.0 from concepts to creativity, Morgan Kaufmann***
- ***CGI programming with Perl, 2do Ed, O'Reilly***
- ***The book of CGI, Prentice Hall***
- ***High Performance Web Site, O'Reilly***
- ***Head First Servlets and JSP, O'Reilly***
- ***http://cri.ch/linux/docs/sk0007.html, Sven Knispel***
- ***http://httpd.apache.org/docs/2.0/howto/cgi.html***

25