PERL - CGI PROGRAMMING

What is CGI ?

- A Common Gateway Interface, or CGI, is a set of standards that defines how information is exchanged between the web server and a custom script.
- The CGI specifications are currently maintained by the NCSA and NCSA defines CGI is as follows −

  The Common Gateway Interface, or CGI, is a standard for external gateway programs to interface with information servers such as HTTP servers.
- The current version is CGI/1.1 and CGI/1.2 is under progress.

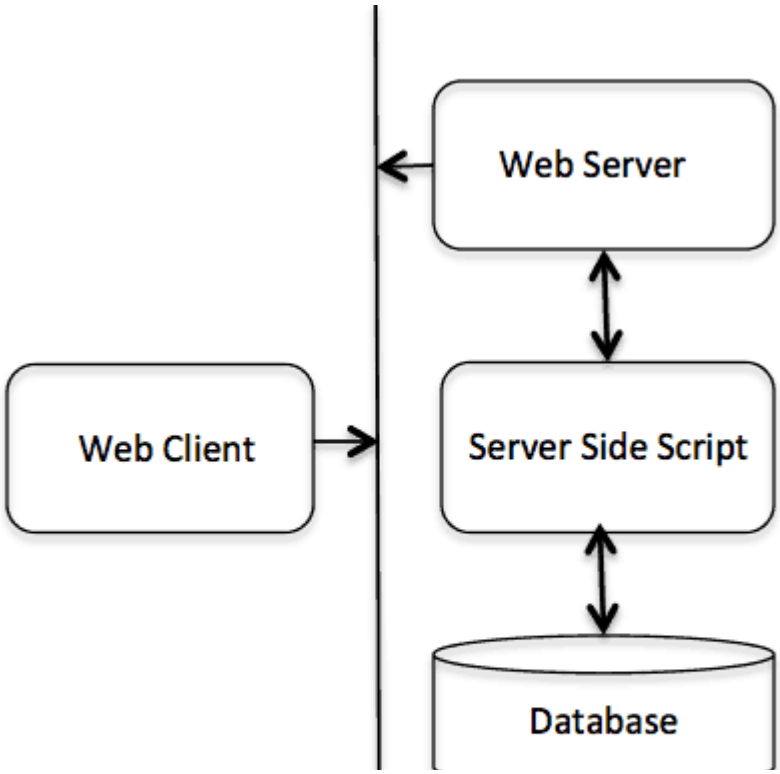
Web Browsing

To understand the concept of CGI, lets see what happens when we click a hyper link available on a web page to browse a particular web page or URL.

- Your browser contacts web server using HTTP protocol and demands for the URL, i.e., web page filename.
- Web Server will check the URL and will look for the filename requested. If web server finds that file then it sends the file back to the browser without any further execution otherwise sends an error message indicating that you have requested a wrong file.
- Web browser takes response from web server and displays either the received file content or an error message in case file is not found.

However, it is possible to set up HTTP server in such a way so that whenever a file in a certain directory is requested that file is not sent back; instead it is executed as a program, and whatever that program outputs as a result, that is sent back for your browser to display. This can be done by using a special functionality available in the web server and it is called **Common Gateway Interface** or CGI and such programs which are executed by the server to produce final result, are called CGI scripts. These CGI programs can be a PERL Script, Shell Script, C or C++ program, etc.

CGI Architecture Diagram



Web Server Support and Configuration

Before you proceed with CGI Programming, make sure that your Web Server supports CGI functionality and it is configured to handle CGI programs. All the CGI programs to be executed by the web server are kept in a pre-configured directory. This directory is called CGI directory and by convention it is named as /cgi-bin. By convention Perl CGI files will have extension as **.cgi**.

First CGI Program

Here is a simple link which is linked to a CGI script called hello.cgi. This file has been kept in **/cgi-bin/** directory and it has the following content. Before running your CGI program, make sure you have change mode of file using **chmod 755 hello.cgi** UNIX command.

```
#!/usr/bin/perl

print "Content-type:text/html\r\n\r\n";
print '<html>';
print '<head>';
print '<title>Hello Word - First CGI Program</title>';
print '</head>';
print '<body>';
print '<h2>Hello Word! This is my first CGI program</h2>';
print '</body>';
print '</html>';

1;
```

Now if you click **hello.cgi** link then request goes to web server who search for hello.cgi in /cgi-bin directory, execute it and whatever result got generated, web server sends that result back to the web browser, which is as follows −

```
Hello Word! This is my first CGI program
```

This hello.cgi script is a simple Perl script which is writing its output on STDOUT file, i.e., screen. There is one important and extra feature available which is first line to be printed **Content-type:text/html\r\n\r\n**. This line is sent back to the browser and specifies the content type to be displayed on the browser screen. Now you must have understood basic concept of CGI and you can write many complicated CGI programs using Perl. This script can interact with any other external system also to exchange information such as a database, web services, or any other complex interfaces.

Understanding HTTP Header

The very first line **Content-type:text/html\r\n\r\n** is a part of HTTP header, which is sent to the browser so that browser can understand the incoming content from server side. All the HTTP header will be in the following form −

```
HTTP Field Name: Field Content
```

For Example −

```
Content-type:text/html\r\n\r\n
```

There are few other important HTTP headers, which you will use frequently in your CGI Programming.

Header	Description
Content-type: String	A MIME string defining the format of the content being returned. Example is Content-type:text/html
Expires: Date String	The date when the information becomes invalid. This should be used by the browser to decide when a page needs to be refreshed. A valid date string should be in the format 01 Jan 1998 12:00:00 GMT.
Location: URL String	The URL that should be returned instead of the URL requested. You can use this filed to redirect a request to any other location.
Last-modified: String	The date of last modification of the file.
Content-length: String	The length, in bytes, of the data being returned. The browser uses this value to report the estimated download time for a file.
Set-Cookie: String	Set the cookie passed through the *string*

CGI Environment Variables

All the CGI program will have access to the following environment variables. These variables play an important role while writing any CGI program.

Variable Names	Description
CONTENT_TYPE	The data type of the content. Used when the client is sending attached content to the server. For example file upload, etc.
CONTENT_LENGTH	The length of the query information. It's available only for POST requests.
HTTP_COOKIE	Returns the set cookies in the form of key & value pair.
HTTP_USER_AGENT	The User-Agent request-header field contains information about the user agent originating the request. Its name of the web browser.
PATH_INFO	The path for the CGI script.
QUERY_STRING	The URL-encoded information that is sent with GET method request.
REMOTE_ADDR	The IP address of the remote host making the request. This can be useful for logging or for authentication purpose.
REMOTE_HOST	The fully qualified name of the host making the request. If this information is not available then REMOTE_ADDR can be used to get IR address.
REQUEST_METHOD	The method used to make the request. The most common methods are GET and POST.
SCRIPT_FILENAME	The full path to the CGI script.
SCRIPT_NAME	The name of the CGI script.
SERVER_NAME	The server's hostname or IP Address.
SERVER_SOFTWARE	The name and version of the software the server is running.

Here is a small CGI program to list down all the CGI variables supported by your Web server. Click this link to see the result Get Environment

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";
print "<font size=+1>Environment</font>\n";
foreach (sort keys %ENV)
{
  print "<b>$_</b>: $ENV{$_}<br>\n";
}

1;
```

Raise a "File Download" Dialog Box?

Sometime it is desired that you want to give option where a user will click a link and it will pop up a "File Download" dialogue box to the user instead of displaying actual content. This is very easy and will be achieved through HTTP header.

This HTTP header will be different from the header mentioned in previous section. For example, if you want to make a **FileName** file downloadable from a given link then it's syntax will be as follows −

```
#!/usr/bin/perl

# HTTP Header
print "Content-Type:application/octet-stream; name=\"FileName\"\r\n";
print "Content-Disposition: attachment; filename=\"FileName\"\r\n\n";

# Actual File Content will go hear.
open( FILE, "<FileName" );
while(read(FILE, $buffer, 100) )
{
   print("$buffer");
}
```

GET and POST Methods

You must have come across many situations when you need to pass some information from your browser to the web server and ultimately to your CGI Program handling your requests. Most frequently browser uses two methods to pass this information to the web server. These methods are **GET** Method and **POST** Method. Let's check them one by one.

Passing Information using GET Method

The GET method sends the encoded user information appended to the page URL itself. The page and the encoded information are separated by the ? character as follows −

```
http://www.test.com/cgi-bin/hello.cgi?key1=value1&key2=value2
```

The GET method is the defualt method to pass information from a browser to the web server and it produces a long string that appears in your browser's Location:box. You should never use GET method if you have password or other sensitive information to pass to the server. The GET method has size limitation: only 1024 characters can be passed in a request string.

This information is passed using **QUERY_STRING** header and will be accessible in your CGI Program through QUERY_STRING environment variable which you can parse and use in your CGI program.

You can pass information by simply concatenating key and value pairs alongwith any URL or you can use HTML <FORM> tags to pass information using GET method.

Simple URL Example: Get Method

Here is a simple URL which will pass two values to hello_get.cgi program using GET method.

http://www.tutorialspoint.com/cgi-bin/hello_get.cgi?first_name=ZARA&last_name=ALI

Below is **hello_get.cgi** script to handle input given by web browser.

```
#!/usr/bin/perl

local ($buffer, @pairs, $pair, $name, $value, %FORM);
# Read in text
$ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
if ($ENV{'REQUEST_METHOD'} eq "GET")
{
   $buffer = $ENV{'QUERY_STRING'};
}
# Split information into name/value pairs
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
   ($name, $value) = split(/=/, $pair);
   $value =~ tr/+/ /;
   $value =~ s/%(..)/pack("C", hex($1))/eg;
   $FORM{$name} = $value;
}
$first_name = $FORM{first_name};
$last_name  = $FORM{last_name};

print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>Hello - Second CGI Program</title>";
print "</head>";
print "<body>";
print "<h2>Hello $first_name $last_name - Second CGI Program</h2>";
print "</body>";
print "</html>";

1;
```

Simple FORM Example: GET Method

Here is a simple example, which passes two values using HTML FORM and submit button. We are going to use the same CGI script hello_get.cgi to handle this input.

```
<FORM action="/cgi-bin/hello_get.cgi" method="GET">
First Name: <input type="text" name="first_name">  <br>

Last Name: <input type="text" name="last_name">
<input type="submit" value="Submit">
</FORM>
```

Here is the actual output of the above form coding. Now you can enter First and Last Name and then click submit button to see the result.

First Name:

Last Name:

Passing Information using POST Method

A more reliable method of passing information to a CGI program is the **POST** method. This packages the information in exactly the same way as GET methods, but instead of sending it as a text string after a **?** in the URL, it sends it as a separate message as a part of HTTP header. Web server provides this message to the CGI script in the form of the standard input.

Below is the modified **hello_post.cgi** script to handle input given by the web browser. This script will handle GET as well as POST method.

```
#!/usr/bin/perl
```

```
local ($buffer, @pairs, $pair, $name, $value, %FORM);
# Read in text
$ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
if ($ENV{'REQUEST_METHOD'} eq "POST")
{
   read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
}else {
   $buffer = $ENV{'QUERY_STRING'};
}
# Split information into name/value pairs
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
   ($name, $value) = split(/=/, $pair);
   $value =~ tr/+/ /;
   $value =~ s/%(..)/pack("C", hex($1))/eg;
   $FORM{$name} = $value;
}
$first_name = $FORM{first_name};
$last_name  = $FORM{last_name};

print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>Hello - Second CGI Program</title>";
print "</head>";
print "<body>";
print "<h2>Hello $first_name $last_name - Second CGI Program</h2>";
print "</body>";
print "</html>";

1;
```

Let us take again same example as above, which passes two values using HTML FORM and submit button. We are going to use CGI script hello_post.cgi to handle this input.

```
<FORM action="/cgi-bin/hello_post.cgi" method="POST">
First Name: <input type="text" name="first_name">  <br>

Last Name: <input type="text" name="last_name">

<input type="submit" value="Submit">
</FORM>
```

Here is the actual output of the above form coding, You enter First and Last Name and then click submit button to see the result.

First Name:

Last Name:

Passing Checkbox Data to CGI Program

Checkboxe are used when more than one option is required to be selected. Here is an example HTML code for a form with two checkboxes.

```
<form action="/cgi-bin/checkbox.cgi" method="POST" target="_blank">
<input type="checkbox" name="maths" value="on"> Maths
<input type="checkbox" name="physics" value="on"> Physics
<input type="submit" value="Select Subject">
</form>
```

The result of this code is the following form −

Maths Physics

Below is **checkbox.cgi** script to handle input given by web browser for radio button.

```
#!/usr/bin/perl

local ($buffer, @pairs, $pair, $name, $value, %FORM);
# Read in text
$ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
if ($ENV{'REQUEST_METHOD'} eq "POST")
{
   read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
}else {
   $buffer = $ENV{'QUERY_STRING'};
}
# Split information into name/value pairs
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
   ($name, $value) = split(/=/, $pair);
   $value =~ tr/+/ /;
   $value =~ s/%(..)/pack("C", hex($1))/eg;
   $FORM{$name} = $value;
}
if( $FORM{maths} ){
   $maths_flag ="ON";
}else{
   $maths_flag ="OFF";
}
if( $FORM{physics} ){
   $physics_flag ="ON";
}else{
   $physics_flag ="OFF";
}

print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>Checkbox - Third CGI Program</title>";
print "</head>";
print "<body>";
print "<h2> CheckBox Maths is : $maths_flag</h2>";
print "<h2> CheckBox Physics is : $physics_flag</h2>";
print "</body>";
print "</html>";

1;
```

Passing Radio Button Data to CGI Program

Radio Buttons are used when only one option is required to be selected. Here is example HTML code for an form with two radio button −

```
<form action="/cgi-bin/radiobutton.cgi" method="POST" target="_blank">
<input type="radio" name="subject" value="maths"> Maths
<input type="radio" name="subject" value="physics"> Physics
<input type="submit" value="Select Subject">
</form>
```

The result of this code is the following form:

Maths Physics

Below is **radiobutton.cgi** script to handle input given by the web browser for radio button.

```
#!/usr/bin/perl

local ($buffer, @pairs, $pair, $name, $value, %FORM);
# Read in text
$ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
if ($ENV{'REQUEST_METHOD'} eq "POST")
{
   read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
}else {
   $buffer = $ENV{'QUERY_STRING'};
}
# Split information into name/value pairs
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
   ($name, $value) = split(/=/, $pair);
   $value =~ tr/+/ /;
   $value =~ s/%(..)/pack("C", hex($1))/eg;
   $FORM{$name} = $value;
}
$subject = $FORM{subject};

print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>Radio - Fourth CGI Program</title>";
print "</head>";
print "<body>";
print "<h2> Selected Subject is $subject</h2>";
print "</body>";
print "</html>";

1;
```

Passing Text Area Data to CGI Program

A textarea element is used when multiline text has to be passed to the CGI Program. Here is an example HTML code for a form with a TEXTAREA box −

```
<form action="/cgi-bin/textarea.cgi" method="POST" target="_blank">
<textarea name="textcontent" cols=40 rows=4>
Type your text here...
</textarea>
<input type="submit" value="Submit">
</form>
```

The result of this code is the following form −

Type your text here...

Below is the **textarea.cgi** script to handle input given by the web browser.

```
#!/usr/bin/perl

local ($buffer, @pairs, $pair, $name, $value, %FORM);
# Read in text
$ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
if ($ENV{'REQUEST_METHOD'} eq "POST")
{
   read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
}else {
   $buffer = $ENV{'QUERY_STRING'};
}
```

```
# Split information into name/value pairs
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
   ($name, $value) = split(/=/, $pair);
   $value =~ tr/+/ /;
   $value =~ s/%(..)/pack("C", hex($1))/eg;
   $FORM{$name} = $value;
}
$text_content = $FORM{textcontent};

print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>Text Area - Fifth CGI Program</title>";
print "</head>";
print "<body>";
print "<h2> Entered Text Content is $text_content</h2>";
print "</body>";
print "</html>";

1;
```

Passing Drop Down Box Data to CGI Program

A drop down box is used when we have many options available but only one or two will be selected. Here is example HTML code for a form with one drop down box.

```
<form action="/cgi-bin/dropdown.cgi" method="POST" target="_blank">
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
<input type="submit" value="Submit">
</form>
```

The result of this code is the following form −

Maths

Below is the **dropdown.cgi** script to handle input given by web browser.

```
#!/usr/bin/perl

local ($buffer, @pairs, $pair, $name, $value, %FORM);
# Read in text
$ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
if ($ENV{'REQUEST_METHOD'} eq "POST")
{
   read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
}else {
   $buffer = $ENV{'QUERY_STRING'};
}
# Split information into name/value pairs
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
   ($name, $value) = split(/=/, $pair);
   $value =~ tr/+/ /;
   $value =~ s/%(..)/pack("C", hex($1))/eg;
   $FORM{$name} = $value;
}
$subject = $FORM{dropdown};

print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
```

```
print "<title>Dropdown Box - Sixth CGI Program</title>";
print "</head>";
print "<body>";
print "<h2> Selected Subject is $subject</h2>";
print "</body>";
print "</html>";

1;
```

Using Cookies in CGI

HTTP protocol is a stateless protocol. But for a commercial website it is required to maintain session information among different pages. For example one user registration ends after transactions which spans through many pages. But how to maintain user's session information across all the web pages?

In many situations, using cookies is the most efficient method of remembering and tracking preferences, purchases, commissions, and other information required for better visitor experience or site statistics.

How It Works

Your server sends some data to the visitor's browser in the form of a cookie. The browser may accept the cookie. If it does, it is stored as a plain text record on the visitor's hard drive. Now, when the visitor arrives at another page on your site, the cookie is available for retrieval. Once retrieved, your server knows/remembers what was stored.

Cookies are a plain text data record of 5 variable-length fields −

- **Expires:** The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
- **Domain:** The domain name of your site.
- **Path:** The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
- **Secure:** If this field contains the word "secure" then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
- **Name=Value:** Cookies are set and reviewed in the form of key and value pairs.

Setting up Cookies

It is very easy to send cookies to browser. These cookies will be sent along with the HTTP Header. Assuming you want to set UserID and Password as cookies. So it will be done as follows −

```
#!/usr/bin/perl

print "Set-Cookie:UserID=XYZ;\n";
print "Set-Cookie:Password=XYZ123;\n";
print "Set-Cookie:Expires=Tuesday, 31-Dec-2007 23:12:40 GMT";\n";
print "Set-Cookie:Domain=www.tutorialspoint.com;\n";
print "Set-Cookie:Path=/perl;\n";
print "Content-type:text/html\r\n\r\n";
...........Rest of the HTML Content goes here....
```

Here we used **Set-Cookie** HTTP header to set cookies. It is optional to set cookies attributes like Expires, Domain, and Path. It is important to note that cookies are set before sending magic line **"Content-type:text/html\r\n\r\n**.

Retrieving Cookies

It is very easy to retrieve all the set cookies. Cookies are stored in CGI environment variable HTTP_COOKIE and they will have following form.

```
key1=value1;key2=value2;key3=value3....
```

Here is an example of how to retrieve cookies.

```
#!/usr/bin/perl
$rcvd_cookies = $ENV{'HTTP_COOKIE'};
@cookies = split /;/, $rcvd_cookies;
foreach $cookie ( @cookies ){
   ($key, $val) = split(/=/, $cookie); # splits on the first =.
   $key =~ s/^\s+//;
   $val =~ s/^\s+//;
   $key =~ s/\s+$//;
   $val =~ s/\s+$//;
   if( $key eq "UserID" ){
      $user_id = $val;
   }elsif($key eq "Password"){
      $password = $val;
   }
}
print "User ID  = $user_id\n";
print "Password = $password\n";
```

This will produce the following result, provided above cookies have been set before calling retrieval cookies script.

```
User ID = XYZ
Password = XYZ123
```

CGI Modules and Libraries

You will find many built-in modules over the internet which provides you direct functions to use in your CGI program. Following are the important once.

- CGI Module
- Berkeley cgi-lib.pl