Copyright © tutorialspoint.com

PERL and CGI Tutorial

Previous Page | Next Page

Advertisements

What is CGI ?

- The Common Gateway Interface, or CGI, is a set of standards that define how information is exchanged between the web server and a custom script.
- The CGI specs 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.

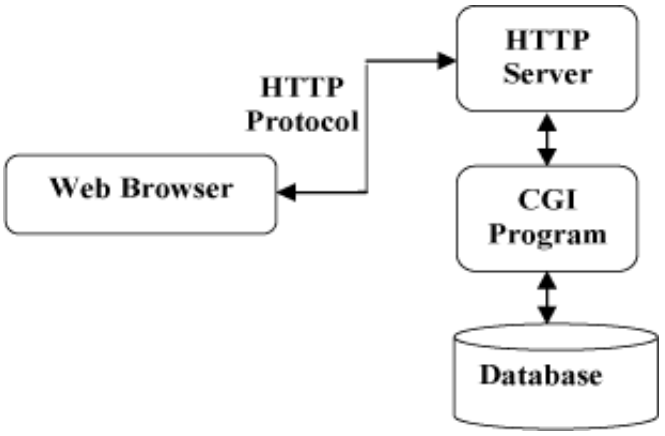
Web Browsing

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

- Your browser contacts the HTTP web server and demand for the URL ie. filename.
- Web Server will parse the URL and will look for the filename in if it finds that file then sends back to the browser 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 or error message.

However, it is possible to set up the HTTP server 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 is sent back for your browser to display. This function is called the Common Gateway Interface or CGI and the programs 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 & Configuration

Before you proceed with CGI Programming, make sure that your Web Server supports CGI and it is configured to handle CGI Programs. All the CGI Programs be executed by the HTTP 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 extention as **.cgi**.

First CGI Program

Here is a simple link which is linked to a CGI script called hello.cgi. This file is being kept in /cgi-bin/ directory and it has following content. Before running your CGI program make sure you have chage 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;

If you click hello.cgi then this produces following output:
```

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 ie. 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 specifiy the content type to be displayed on the browser screen. Now you must have undertood basic concept of CGI and you can write many complicated CGI programs using PERL. This script can interact with any other exertnal system also to exchange information such as RDBMS.

HTTP Header

The line **Content-type:text/html\r\n\r\n** is part of HTTP header which is sent to the browser to understand the content. 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
	A MIME string defining the format of the file being returned. Example

Content-type: String	is Content-type:text/html
Expires: Date String	The date 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 file.
Last-modified: String	The date of last modification of the resource.
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 Name	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	Return 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 small CGI program to list out all the CGI variables. 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;
```

How To Raise a "File Download" Dialog Box ?

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

This HTTP header will be different from the header mentioned in previous section.

For example,if you want make a **FileName** file downloadable from a given link then its 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 web server and ultimately to your CGI Program. Most frequently browser uses two methods two pass this information to web server. These methods are GET Method and POST Method.

Passing Information using GET method:

The GET method sends the encoded user information appended to the page request. 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 browser to web server and it

produces a long string that appears in your browser's Location:box. Never use the GET method if you have password or other sensitive information to pass to the server. The GET method has size limtation: only 1024 characters can be 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

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 same CGI script hello_get.cgi to handle this imput.

```
<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, You enter First and Last Name and then click submit button to see the result.

First Name:

Last Name: Submit

Passing Information using POST method:

A generally 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. This message comes into the CGI script in the form of the standard input.

Below is hello_post.cgi script to handle input given by 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 examle 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 imput.

```
<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, You enter First and Last Name and then click submit button to see the result.

First Name:

Last Name: Submit

Passing Checkbox Data to CGI Program

Checkboxes are used when more than one option is required to be selected.

Here is 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 Select Subject

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 a 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

Below is radiobutton.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;
    }
    $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

TEXTAREA element is used when multiline text has to be passed to the CGI Program.

Here is 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...

Submit

Below is textarea.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;
    }
    $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

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

Below is 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 completing 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 retrviewed in the form of key and value pairs.

Setting up Cookies

This is very easy to send cookies to browser. These cookies will be sent along with 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....
```

From this example you must have understood how to set cookies. We use **Set-Cookie** HTTP header to set cookies.

Here it is optional to set cookies attributes like Expires, Domain, and Path. It is notable that cookies are set before sending magic line **"Content-type:text/html\r\n\r\n**.

Retrieving Cookies

This 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 retrieving 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 following result
User ID = XYZ
Password = XYZ123
```

CGI Modules and Libraries

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

- CGI Module
- Berkeley cgi-lib.pl

Previous Page | Next Page

Copyright © tutorialspoint.com