Beginning CGI Programming in Perl

In this section we will lay the foundation for CGI script development.

We will introduce general CGI programming concepts relating to CGI output but then focus on Perl programming.

Specifically we will develop a very simple Perl program and see how to run it on a Macintosh and UNIX platform.




CGI Script Output

We have already mentioned that CGI scripts must adhere to standard input and output mechanism

- The **Interface** between browser and server
- Part of HTTP Protocol

For the moment we will not worry about input to a CGI script.




CGI Script Output Format

In whatever language a CGI script is programmed it **MUST** send information back in the following format:

- **The Output Header**
- **A Blank Line**
- **The Output Data**

NOTE: Between the Header and Data there **MUST** be a blank line.




CGI Output Header

- A browser can accept input in a variety of forms.
- Depending on the specified form it will call different mechanisms to display the data.
- The output header of a CGI script must specify an output type to tell the server and eventually browser how to proceed with the rest of the CGI output.




Three forms of Header Type

There are 3 forms of Header Type:

- `Content-Type`
- `Location`
- `Status`

`Content-Type` is the most popular type.

- We now consider this further.
- We will meet the other types later.




Content-Types

The following are common formats/content-types (there are a few others):

Format	Content-Type
HTML	`text/html`
Text	`text/plain`
Gif	`image/gif`
JPEG	`image/jpeg`
Postscript	`application/postscript`
MPEG	video/mpeg




Declaring `Content-Type`

To declare the `Content-Type` your CGI script must output:

`Content-Type:` **content-type specification**

Typically the `Content-Type` will be declared to produce HTML.

So the first line of our CGI script (for most of our examples) will look this:

```
Content-Type:  text/html
```





CGI Output Data

Depending on the `Content-Type` defined the data that follows the header declaration will vary:

- If it is **HTML** that follows then the CGI script must output **standard HTML syntax**.

Example: To produce a Web page that the server sends to a browser with a simple line of text "Hello World!" . A **CGI script must output**:

```
Content-Type: text/html


<html>
<head>
<title>Hello, world!</title>
</head>
<body>
<h1>Hello, world!</h1>
</body>
</html>
```

Now let us see how we write and display in a Browser this CGI script in Perl




A First Perl CGI Script

Let us now look at how we write our first perl program that will be used as a CGI script.

We will learn three main things in here:

- The basic **format** of Perl CGI program
- How to **comment** Perl programs
- One Perl function `print` — which outputs data:
  - As a CGI Perl Program — Data sent to browser
  - As a stand alone Perl Program (Non- CGI) — Data sent to standard output (default: terminal window)




Format of a Perl program

Every Perl program **MUST** obey the following format:

- A first line consisting of:

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

- The rest of the program consisting of **legal** Perl syntax and commands

Strictly speaking the first line is **only required for running Perl programs on UNIX machines**.

- Since that is the intended destination of most of our Perl/CGI scripts.
- It is a good idea to **make this** the first line of every perl program.

What is the purpose of this first line?

The first line declaration has two purposes:

- It indicates that the program is a Perl script.
- It tells UNIX how to run Perl.
  Do not worry too much about this last fact — it basically specifies where in the directory hierarchy the perl interpreter program resides.
- **It MUST** be typed exactly as above to run School's UNIX/MAC OS X systems.
- The exact location **may vary** on other systems.

The first line is actually a **comment**

- Albeit a very special type of comment




Comments in Perl

It is good practice to comment all your programs (whatever the language (HTML, Perl, Java, ...) — **Suitable comments serve all programmers well**.

In Perl comments are easy:

- The # symbol indicates a comment.
- The remainder of the line is regarded as a comment
- **DO NOT** put any Perl code after a # symbol until you type a carriage return — as this will always be ignored by Perl

So a simple comment in Perl might be:

```
# hello.pl - My first Perl CGI program
```





Output from Perl

To output from a Perl script you use the `print` statement:

- The `print` statement takes a string (delimited by `"... "`) argument which it outputs.
- Similat to Java (and especially C) the string argument formats output.
  - You can control how the output looks from a single `print` statement.
  - The `\n` character indicates that a newline is required at that point in the text.
  - We will introduce further aspects of the `print` statement later.




First Line Output of a CGI script in Perl

For Example, The first line of our CGI script must be

- "`Content-Type:  text/html`" and
- The `print` statement must have 2 `\n` characters:
  - One to terminate the current line, and
  - The second to produce the require blank line between CGI header and data.
- So our completer Perl line looks like this:

```
print "Content-Type: text/html\n\n";
```





Finally — Our complete script

Recall that our Perl CGI script must output the header and HTML code and must begin with a special first line.

Our complete first program (with nice comments) is a follows:

```
#!/usr/bin/perl
# hello.pl - My first Perl CGI program

print "Content-Type: text/html\n\n";
# Note there is a newline between
# this header and Data

# Simple HTML code follows

print "<html> <head>\n";
print "<title>Hello, world!</title>";
print "</head>\n";
print "<body>\n";
print "<h1>Hello, world!</h1>\n";
print "</body> </html>\n";
```





Writing, Creating and Running CGI Perl Scripts

We now know what a (simple) Perl script looks like.

Let us now look at how we create and run Perl scripts.

We will look at how we create Perl Scripts on a Macintosh or UNIX and how we run Perl scripts as standalone and CGI scripts on Macintosh and UNIX.




Writing/Creating Perl Scripts

Perl Scripts are basically text files with special perl syntax embedded in the text.

Therefore any **text editor** can be used to create and edit you Perl files.

On the Macintosh Computer **BBEdit Lite** is the recommended text editor.

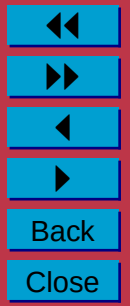



Running Perl on Mac OS X/UNIX/LINUX Command Line

- Simply fire up a terminal window or
- Open Telnet connection to UNIX machine
- Make sure the Perl script is executable
  - The UNIX command:

    ```
    chmod +x myperl.pl
    ```

    achieves this.
  - To see whether a file is **execuable** use UNIX command `ls -l`,



`ls -l` To See if File is Executable Example

E.G.:

```
ls -l myperl.pl
```

You should see something like:

```
-rwxr-xr-x 1 dave staff 356 Nov 19 2003 myperl.pl
```

Look for the `x` in the User, Group and/or All file permissions

- Either simply type the file name from the command:

  `myperl.pl`, or

- Run the perl interpreter, `perl`, with the file name:

  ```
  perl myperl.pl
  ```





Test Perl Script Locally First

- If you run perl scripts from the command line they **DO NOT** function as a CGI script
- **However you can verify that the scripts syntax is correct**
  - and save wasted file copying to web server
- **Possibly you can verify that the scripts output is correct**
  - by manually viewing the script output on the command line
  - E.G. Basic HTML syntax
  - and save wasted file copying to web server




Running Perl on School's UNIX/LINUX Web Server

We assume that a Perl Script has been created and tested on a Macintosh Locally.

To run a CGI Perl script on UNIX, Simply:

- Samba File Copy or FTP (use **Fetch**) the Perl Script to the appropriate **cgi-bin** directory on UNIX (`project` or `public`).
- Put associated HTML file in appropriate **html** directory on UNIX (`project` or `public`).
- Reference Perl script either via
  - a FORM — Make sure URL is the Correct UNIX URL
  - Directly with a URL
- The URL is either
  `http://www.cs.cf.ac.uk/project/A.B.Surname/cgi-bin/file.pl`,
  or
  `http://www.cs.cf.ac.uk/user/A.B.Surname/cgi-bin/file.pl`.