7.7 CGI - Common Gateway Interface

CGI is a standard for helping web servers run external programs and return *dynamic* web pages.

For example, a simple dynamic web page might return the current date and time, calculated by running the '**date**' program, and formatting the results as a web page. The following script shows the idea:

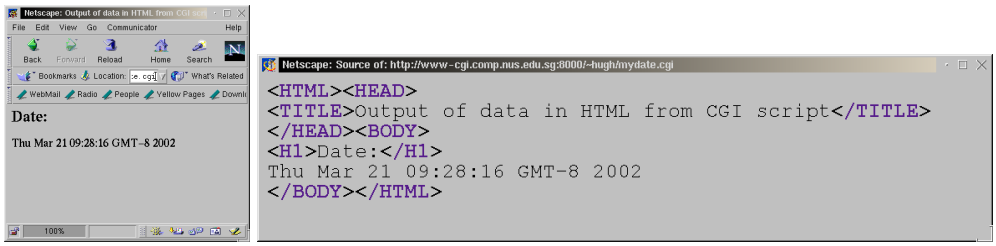
CODE LISTING **mydate.cgi**

```
#!/bin/sh

cat <<EOM1
Content-type: text/html

<HTML><HEAD>
<TITLE>Output of data in HTML from CGI script</TITLE>
</HEAD><BODY>
<H1>Date:</H1>
EOM1
date
cat <<EOM2
</BODY></HTML>
EOM2
```

When this script is placed in the directory **public_cgi** in your home directory on one of the UNIX systems, then you may refer to

http://www-cgi.comp.nus.edu.sg:8000/~yourid/mydate.cgi

and you will get the following display:



This is of course a trivial example, but shows the fundamental idea of running a script, to get dynamic content in a web page. In this case, the script is not passed any data from the client browser - it just runs the shell **/bin/sh** and the **date** program on the server.

Note that there is no requirement for your CGI program to be a shell script. You may use any suitable scripting language, or compiled programs, and **perl** is very commonly used in this role. The most important thing to remember is that whatever your CGI program does, it should not take too long to process.

7.7.1 CGI environment variables

To pass parameters to CGI programs, environment variables are used. The following **`perl`** script can display all the enviroment variables passed to a CGI program:

CODE LISTING **env.cgi**

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

print "Content-type: text/html\n\n";
print <<EndOfHTML;
<html><head><title>Print Environment</title></head>
<body>
EndOfHTML

foreach $key (sort(keys %ENV)) {
   print "$key = $ENV{$key}<br>\n";
}

print "</body></html>";
```

When this script is run, we get something like the following. This may go some way towards explaining to you how some systems know which web browser you are using!

```
DOCUMENT_ROOT = /usr/local/apache/htdocs
GATEWAY_INTERFACE = CGI/1.1
HTTP_ACCEPT = image/gif, image/x-
xbitmap, image/jpeg, image/pjpeg, image/png, */*
HTTP_ACCEPT_CHARSET = iso-8859-1,*,utf-8
HTTP_ACCEPT_ENCODING = gzip
HTTP_ACCEPT_LANGUAGE = en
HTTP_CONNECTION = Keep-Alive
HTTP_HOST = www-cgi.comp.nus.edu.sg:8000
HTTP_REFERER = http://www-cgi.comp.nus.edu.sg:8000/~hugh/
HTTP_USER_AGENT = Mozilla/4.79 [en] (X11; U; Linux 2.2.16 i686)
PATH = /usr/local/bin:/usr/bin:/bin:/usr/local/php/bin
QUERY_STRING =
REMOTE_ADDR = 137.132.90.155
REMOTE_HOST = dhcp-hugh.ddns.comp.nus.edu.sg
REMOTE_PORT = 3343
REQUEST_METHOD = GET
REQUEST_URI = /~hugh/env.cgi
SCRIPT_FILENAME = /home/staff/hugh/public_cgi/env.cgi
SCRIPT_NAME = /~hugh/env.cgi
SERVER_ADDR = 137.132.90.7
SERVER_ADMIN = websp@comp.nus.edu.sg
SERVER_NAME = www-cgi.comp.nus.edu.sg
SERVER_PORT = 8000
SERVER_PROTOCOL = HTTP/1.0
SERVER_SOFTWARE = Apache/1.3.19 (Unix) PHP/4.0.5 mod_perl/1.25
TZ = Singapore
```

The bold strings are the names of the environment variables passed to the CGI program. To the right of the name is the value of that environment variable.

7.7.2 CGI forms

CGI is commonly used for processing simple form-based applications. That is, the client display has a form, and the user keys-in, or selects items in the form, which is then submitted to the CGI program for processing. The principal mechanism for passing small form contents to the CGI program uses environment variables, which are passed to the called CGI program.

The form contents are found inside an environment variable called **QUERY_STRING**, as a series of **name/value** pairs. This mechanism is known as the **GET** mechanism, and a typical URL would look like this:

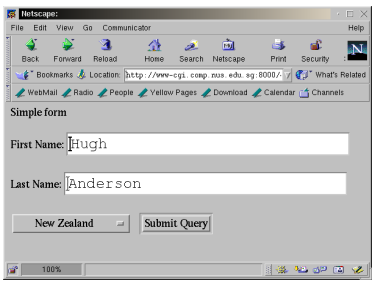
```
http://www-cgi.comp.nus.edu.sg:8000/~yourid/myform.cgi?name1=value1&name2=value2
```

An alternative mechanism is the **POST** mechanism, in which the **STDIN** of the CGI program is used to process the form data. A simple example of a form based web page:

CODE LISTING **form.html**

```
<html><head>Simple form</head>
<body>
<form action="env.cgi" method="GET">
      First Name:  <input type="text" name="First" size=30><p>
      Last Name:  <input type="text" name="Last" size=30><p>
      <select name="Home">
      <option>Singapore    <option>Malaysia
      <option>Indonesia    <option>New Zealand
      <option>The rest of the world!
      </select>
      <input type="submit">
</form>
</body></html>
```

This produces a page that looks like this:



When the form is submitted, the **QUERY_STRING** looks like this:

```
QUERY_STRING = First=Hugh&Last=Anderson&Home=New+Zealand
```

Within a CGI program, this series of name-value pairs may be used to return a dynamic web page based on this form data. **Perl** is a particularly useful language to use in this context, as it has powerful operators form managing strings, and the **QUERY_STRING** can be **split** quickly into its component parts. There are security issues with unrestricted CGI programs - since they run powerful programs (like perl and csh) with arbitrary parameters, they may be a source of (hacker) intrusion. It is for this reason that CGI usage is restricted here at NUS.

7.8 PHP

PHP is a server-side scripting language that is embedded in your web pages. It looks very like standard HTML scripts, but when a client browser queries a PHP enhanced web page, the server automatically interprets the PHP, and then sends back an ordinary HTML page. There are no enhancements needed for browsers to access PHP web pages.

The two tags **<?php** and **?>** start and end a PHP script, and identify a PHP code segment. The PHP code itself is a reasonably powerful programming language similar to Java, C and Perl, with functions, variables and so on.

PHP stands for PHP - Hypertext Preprocessor, a recursive acronym (like GNU - Gnu's Not UNIX), and is a generally useful HTML/server preprocessor. However it is particularly useful if you wish your web pages to access databases. It is common to pair up PHP with MySQL, but PHP is not limited to one database type. It may be used with any of the commercial databases. For example if you wish to use PHP to access a Microsoft SQL server, you can install the ODBC support in the server machine, and access the server directly.

Here is sample PHP code embedded in a PHP-enhanced web page. It shows PHP connection to a MySQL server, selection of a database and an SQL query:

```
<?php
    ...
    mysql_pconnect("host","user","password")
        or die("Unable to connect to SQL server");
    mysql_select_db("dbasename")
        or die("Unable to select database");
    $numguests = mysql_query("SELECT COUNT(*) FROM guests")
        or die("Select Failed!");
    ...
?>
```

This may all be integrated with standard HTML and form-based web pages to construct a GUI. PHP suffers less from the security issue than perl or csh CGI scripts do. After writing this sentence I went to see what the latest security issues with PHP were, and discovered that recently there has been a major loophole discovered in PHP POST upload code:

> *Each of the flaws could allow an attacker to execute arbitrary code on the victim's system.*

The exploit appers to have been disovered before any use of it, so assuming you have a relatively recent installation of PHP, you should in general have less security worries than with CGI scripts.

7.9 Java enhanced

In Section 5.5 we saw a very simple hello-world Java applet inserted in a web page. Here is a little Java applet for a Lissajous figure:

CODE LISTING **Lissajous1.java**

```
/* @(#)Lissajous.java
 *  Original version was written in 0.4 95/04/09
 *  by Hugh Anderson for HotJava browser.
 *
 *  Updated by L. Gladney to Java 1.0 JDK on 4/13/97.
 *
 * Patrick Chan (chan@scndprsn.Eng.Sun.COM ) has suggested that it
 * would be nice if every point  had a different display, so mouse
 * X motion now controls the ratio of frequencies, and mouse Y motion
 * controls the amplitude. */

import java.applet.Applet;
import java.awt.*;

public class Lissajous extends Applet implements Runnable {
    Thread animate=null;
    double pi=3.14159265359;
    int fx=50;
    int fy=100;
    int diffx=0;
    int amp=50,phase=0;                          // amplitude, phase
    int delay = 50;                              // speed set by length
                                                 // of sleep between refreshes
    public void init() {
        resize(200, 200);                        // resize to fixed width,height
    }

    public void paint(Graphics g) {
        int X,Y,YY=0,lastx=0,lasty=0,temp=0,rev=0;
        g.drawRect(0, 0, size().width - 1, size().height - 1);  // outline
        if ( fy < fx ) {                                        // frequency
            temp = fx;
            fx = fy;
            fy = temp;
            rev = 1;
        }
        for (int x = 0 ; x <= 360 ; x += 4) {                   // loop
            X = (int) ( amp*Math.sin( x*2.0*pi/360.0 ));        // x pos
            YY = (x*fy/fx)+phase;
            Y = (int) ( amp*Math.sin( YY*2.0*pi/360.0 ));
            if (x==0) { lastx=X; lasty=Y; }
            if (rev==1) { g.drawLine(lastx+100,lasty+100,X+100,Y+100); }
             else { g.drawLine(lasty+100,lastx+100,Y+100,X+100); }
            lastx=X;
            lasty=Y;
        }
        if ( rev==1 ) {
           temp=fx;
           fx = fy;
           fy = temp;
         }
        phase = YY;
        /* Fix an error ... phase shouldn't increase forever..... */
        if ( phase < 0 ) { phase += 360; };
        if ( phase >=  360 ) { phase -= 360; };
        g.drawString( fx + ":" + fy,10,20);
    }
```

Here is the rest of the code...

CODE LISTING **Lissajous2.java**

```
    public void run() {
        while (true) {
            repaint();
            try { Thread.currentThread().sleep(delay);  // delay
            }
              catch (Exception e) { };
        }
    }

    public void start() {
        if ( animate == null ) {
            animate = new Thread(this);
            animate.start();
        }
    }
    public void stop() {
        if (animate != null ) {
            animate.stop();
            animate = null;
        }
    }

    public boolean mouseDown(Event e, int x, int y ) {
    Graphics gc;
    gc = getGraphics();

        diffx = fx-x;
    System.out.println("Got a mouse event at " + x + "," + y);
        return true;
    }

    public boolean mouseDrag(Event e, int x, int y) {
        fx = x+diffx;
        if ( fx <= 0 ) { fx = 1; };
        amp = y;
        return true;
    }

    public String getAppletInfo() {
                return "Lissajous by Hugh Anderson/Larry Gladney ";
        }

        public String[][] getParameterInfo() {
                String [][] info = {
                        {"delay    ","int     ", "delay, default=50"}
                };
                return info;
        }

}
```

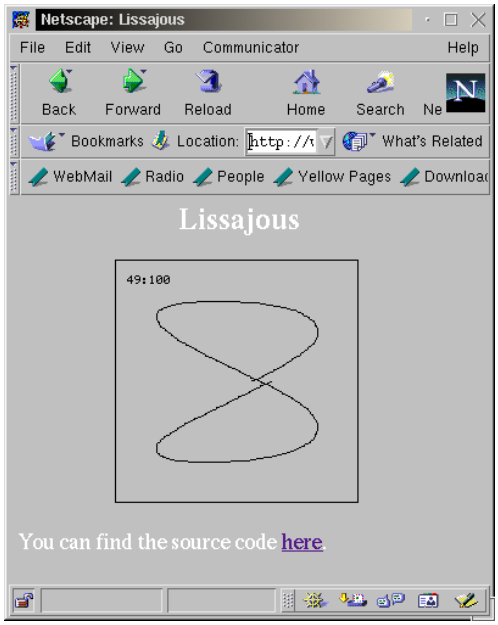
I wrote this code soon after the first Java language was made public, and had forgotten about it until last week, when I went looking for an applet, and found my name on it! This code may be found at

```
http://olddept.physics.upenn.edu/courses/gladney/minicourse/lectures/lecture2.html
```

or locally at

```
http://www.comp.nus.edu.sg/~hugh/Lissajous/Lissajous.html
```

This produces a page that looks like this:



7.10 Summary of topics

In this module, we introduced the following topics:

- Web-based application architectures
- CGI, PHP and Java

Tutorial 6 - questions for week 12 (Mar 27, 2002)

1. (Research) Make up a simple CGI form, similar to the one given on page 71, which uses the **`POST`** method for reading the data, and prints out the three fields.
2. (Research) Make up a simple PHP processed form, similar to the one given on page 71, and which prints out the three fields.
3. Find an example of a site which is using PHP/MySQL with a large database. Give the URL, and a brief note on the site (size of database, type of user interface...).
4. Examine the **`Lissajous.java`** code. What is the function of the **`diffx`** variable?

Further study

- http://php.net