CGI INTERFACE

May 12, 2007

HL

I. General Background on Interface Techniques

To establish CGI C program interface with the user browser, we use the following example to illustrate 4 steps operation.

Example: Perform addition of two integers, the GUI input of the browser will ask a user to enter the first and the second operand, while the CGI program will perform the addition and send the result back to the browser.

Step 1. A HTML page to allow users to input 2 integers and press a button to send the HTTP request to the web server. The HTML file is named as "addition.html".

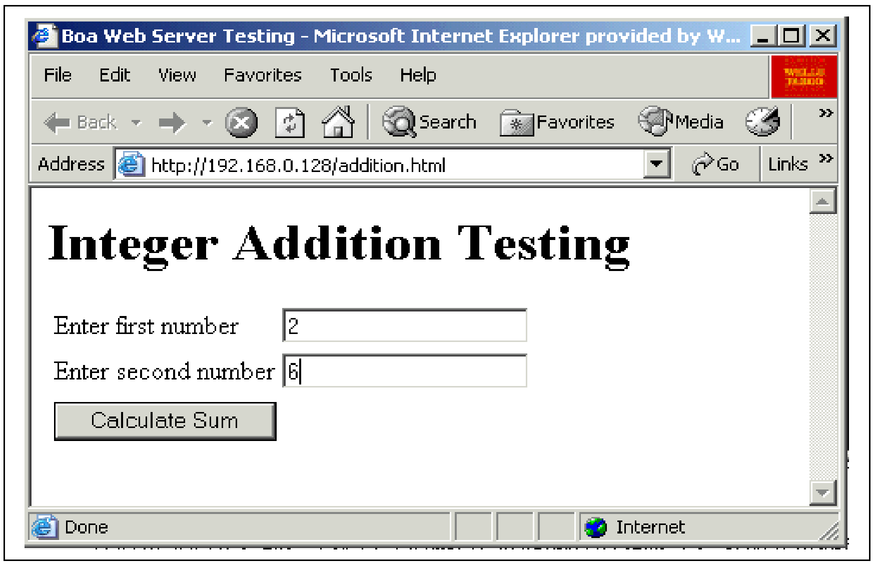
Step 2. A C program properly placed in the /cgi-bin directory has

1. A function to parse out HTTP request header to get the information from the browser.
2. A function to perform the addition.
3. A "printf(…)" in C to embed the results into a HTML page to the web server for responding to the browser. In a CGI enabled process standard input and output are "hijacked" by CGI communication module. As a result, in CGI program, a "printf (…)" C language statement sends output to web server instead of to standard output (OS terminal). Therefore, the HTML page is received by web server when CGI programs are doing "printf". Web server serves as an agent to transfer all the HTTP responses to the associated web clients (browser).

II. Implementation Details

II.1 The HTML File

The HTML file on the web server gives the web browser's GUI experience. The HTML example of CGI addition is shown in Figure 1.


Boa Web Server Testing - Microsoft Internet Explorer provided by W...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://192.168.0.128/addition.html
Go Links
Done
Internet

Integer Addition Testing

Enter first number: 2

Enter second number: 6

Calculate Sum

Figure 1. The GUI of the CGI addition example.

The html file for the GUI is given in Figure 2.

```
<html>
<head>
    <title> Web Server Testing </title>
</head>
<body>
<h1>Integer Addition Testing</h1>
<form action="/cgi-bin/testcgi" method="get">
<table>
    <tr><td>Enter first number</td>
          <td> <input type=text name=number1> </td>
    </tr>
    <tr><td>Enter second number</td>
          <td> <input type=text name=number2> </td>
    </tr>
    <tr><td colspan=2>
          <input type = submit value = "Calculate Sum" >
          </td>
    </tr>
    <input type=hidden name=cmd value="addition">
</table>
</form>
</body>
</html>
```

Figure 2. addition.html

The HTML file demonstrate some key features needed for CGI interface, Table 1 lists these features, keep in mind the browser and CGI interface is accomplished through Web Server via STDIN.

Form	action="…"	`<form action="/cgi-bin/testcgi" method="get">`	attribute
	method="…"	<form action="/cgi-bin/testcgi" method="get">	command
Input	type=…	`<input type=text name=number1>`	Input tag to enter user data; type: text
		<input type = submit value = "Calculate Sum" >	Collect data and posts data
		<input type=hidden name=cmd value="addition">	hidden
	name=…	<input type=text name=number1>	user-defined number
	value=…	<input type=hidden name=cmd value="addition">	Passing variables

Note:

1. The "submit" button is a special "form" element, it
   a. Collects all the data from the "form";
   b. Posts (sends) it to where defined in the "action" portion.
2. A "hidden" HTML field is for passing along variables w/ values from one form to another page.

3. Reference for HTML form: http://www.w3.org/TR/html4/interact/forms.html

2.2. CGI C Program for the Addition Example

Step 1. CGI communicates with the browser via web server as in Figure 3 below.

```
     printf("Content-type: text/html\n\n");
     printf("<html> <head> <title> Web Serve </title>
                           </head>\n");
```

Figure 3. CGI printf(…) to communicate to the browser via web server.

Step 2. Capture the value of the command from the browser as below in Figure 4.

```
     unsigned char *str1,*str2, *cmd;
     cmd = getval((unsigned char *)"cmd");
```

Figure 4. CGI captures input from the browser.

Note the HTML file on the web server use “hidden” attribute to pass variables, “cmd: in this sample program, as shown in Figure 5.

```
<input type=hidden name=cmd value="addition">
```

Figure 5. HTML statement on the web server .

The CGI C program captures parameters passed by HTML page through “getenv(…)” as shown in the following Figure 6.

```
     vstr = (unsigned char *) getenv ( "REQUEST_METHOD");
     if(vstr == NULL)    return 0;
     if(strcmp((const char *)vstr,"POST") == 0)
```

Figure 6. C Code to capture HTML parameters.

Step 3. Capture parameters, first find the length of the parameters as in Figure 7, then capture the parameters from the browser using fgets(…) function in the C stdio library as in Figure 8.

```
         vstr = (unsigned char *) getenv("CONTENT_LENGTH");
         if (vstr == NULL || strlen((const char *)vstr) == 0)
             return 0;
```

Figure 7. C Code to capture the length of the browser’s parameter input.

```
       if(vstr == NULL)
             return 0;
       fgets((char *)vstr,cl+1,stdin);
```

Figure 8. C Code to capture browser’s parameter input, note from the stdin.

III. Appendix C Code

```
#include    <stdlib.h>
#include    <ctype.h>
#include    <string.h>
#include    <stdio.h>
#include    <unistd.h>
#include    <fcntl.h>
#include    <string.h>

#define LINELEN 1024

sruct valinfo
{
    unsigned char *name;
    unsigned char *text;
};

unsigned char *getval(unsigned char *);

static int gotvals=0;
static int nvals=0;
static struct valinfo *vals;
static int hextobin(unsigned char);
static unsigned char  *httpunescape(unsigned char *);
static int getvals(void);

unsigned char *getval(unsigned char *name)
{
    int i;
    If (!gotvals)
    {
        nvals = getvals();
        gotvals = 1;
    }
    if (nvals == 0) return NULL;

    for (i=0;i<nvals;i++)
    {
        if(strcmp((const char *)name, const char *)vals[i].name)==0)
            return vals[i].text;
    }
    return NULL;
}


/* =============== httpunescape =========== */
static unsigned char *httpunescape(unsigned char *sis)
{
    unsigned char *siptr;
    unsigned char *soptr;
    unsigned char *sos;
    sos = (unsigned char *) calloc (
    strlen(char *)sis)+1,sizeof (unsigned char));
    if(sos == NULL)     return NULL;
    soptr = sos;
    siptr = sis;
    while (*siptr)
    {
        if(*siptr == '%')
        {
            int c = 0, i;
            for (i=0;i<2;i++)
            {
                int h;
                siptr++;
                if (*siptr == '\0') break;
                if ((h=hextobin(*siptr)) == -1) break;
                c = c<<4 + h;
            }
            If (i != 2)
            {
                free(sos);
                return NULL;
            }
            *soptr++ = (unsigned char) c;
                } else if (*siptr == '+')
            *soptr++ = ' ';
        else
            *soptr++ = *siptr;
            siptr++;
        }
        *soptr = '\0';

        strcpy ((char *)sis, (const char *)sos);
        free (sos);
        return sis;
}

/* =============== hextobin ================= */
static int  hextobin(unsigned char c)
{
    if(isdigit(c))
        return c-'0';
    else if (isxdigit(c))
        return tolower(c) - 'a' + 10;
    else
        return -1;
}

/* =============== getvals =================== */
static int  getvals(void)
{
    int i;
    int vcnt = 0;
    unsigned char *vstr;
    unsigned char *vptr;
    unsigned char *eptr;
    unsigned char *aptr;

    vstr = (unsigned char *) getenv ( "REQUEST_METHOD");
    if(vstr == NULL)    return 0;
    if(strcmp((const char *)vstr,"POST") == 0)
    {
        int l, cl;
        vstr = (unsigned char *) getenv("CONTENT_LENGTH");
        if (vstr == NULL || strlen((const char *)vstr) == 0)
            return 0;
        if ((cl = atoi((const char *)vstr)) == 0)
```

```
            return 0;
        vstr = (unsigned char *)malloc(cl+2);
        if(vstr == NULL)
            return 0;
        fgets((char *)vstr,cl+1,stdin);
        l = strlen((const char *)vstr);
        if(vstr[l-1] == '\n')
            vstr[l-1]='\0';
    }
    else
    {
        vstr = (unsigned char *) getenv("QUERY_STRING");

        if(vstr == NULL) return 0;
    }
    vptr = vstr;

    while (*vptr)
        if(*vptr++ == '&') vcnt++;

    vcnt++;
    vals = (struct valinfo *)calloc(vcnt,sizeof (struct valinfo));
    if(vals == NULL) return 0;
    vptr = vstr;

    for (i=0;i<vcnt;i++)
    {
        eptr = (unsigned char *)strchr((const char *)vptr,'=');
        aptr = (unsigned char *)strchr((const char *)vptr,'&');

        if (eptr == NULL)
            return 0;
        *eptr = '\0';
        vals[i].name = httpunescape(vptr);
        if (vals[i].name == NULL) return 0;
        if (aptr)
        {
           *aptr = '\0';
           vptr = aptr+1;
        }
        vals[i].text = httpunescape(eptr+1);
        if(vals[i].text == NULL)    return 0;
    }
    return vcnt;
}

/* =============== main =================== */
main ()
{
    int x, y;
    unsigned char *str1,*str2, *cmd;
    cmd = getval((unsigned char *)"cmd");
    printf("Content-type: text/html\n\n");
    printf("<html> <head> <title> Web Serve </title>
                          </head>\n");
    if (strcmp((const char *)cmd, "addition") == 0)
    {
        printf("<body>
                   <h1>Integer Addition Testing Result
                    </h1>\n");
        str1 = getval((unsigned char *)"number1");
        str2 = getval((unsigned char *)"number2");
        if (str1 == NULL || str2 == NULL) {
            printf("<p>Input data error\n");
        } else {

            x = atoi ((const char *)str1);
            y = atoi ((const char *)str2);
            printf("<p>The sum of %d and %d  is %d\n",x,y,x+y);
        }
     } else {
        printf ("<p>Sorry, the request is invalid.\n");
    }
    printf ("</body></html>\n");

    exit (0);
}
```

(END)