

Common Gateway Interface

- CGI
-

Copyright (c) 2013-2015 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Common Gateway Interface

CGI specifies
how the **HTTP server** interfaces
with an external program (**gateway**)

4 interface components

- | | |
|---------------------------|---|
| (a) command line | forms of the command line when the server calls the gateway |
| (b) environment variables | these values are transmitted to the gateway |
| (c) standard input | sent to the gateway |
| (d) standard output | sent to the server |

The Output of CGI

CGI uses
standard output

Content-Type: MIME-type
empty line

MIME standard content types

- *text*
- *image*
- *audio*
- *video*
- *multipart*
- *application*
- *message*

HTTP Transactions

1. Connection

2. Query

3. Processing

4. Response

5. Termination

A query line has 3 parts:

- a query method
- local path of the requested resource
- an HTTP version number

A response line has 3 parts:

- an HTTP version number
- a status code
- a textual description of the status

HTML Forms

Form Handling

Client side languages: Javascript

Server side languages: PHP, JSP

form_handler.php

```
<html>
<body>
<?php
// This will print whatever the user put
$name = $_GET['user'];
echo "Hello, ". $name . "!";
?>
</body>
</html>
```

form.html

```
<html>
<body>
<form action="form_handler" method="GET">
  User Name: <input name="user" type="text" />
  <input type="submit" />
</form>
</body>
</html>
```

form_handler.cpp

```
<html>
<body>

Hello, world!

</body>
</html>
```

HTTP Transactions

A CGI script

- in a special directory : `cgi-bin`
 - obtains URL-encoded **query data**
 - access command-line arguments
 - access environment variables
 - standard output goes to the server
- through standard input for the **POST** method
 - through the **QUERY_STRING** environment var for the **GET** method

- | | |
|----------------|--|
| 1. Connection | 1. process command-line arguments |
| 2. Query | 2. obtain form's input data |
| 3. Processing | 3. process the input data |
| 4. Response | 4. send a response through a standard output |
| 5. Termination | 5. terminate |

Intermediate Nodes

Server specific variables:

SERVER_SOFTWARE: name/version of HTTP server.

SERVER_NAME: host name of the server, may be dot-decimal IP address.

GATEWAY_INTERFACE: CGI/version.

Request specific variables:

SERVER_PROTOCOL: HTTP/version.

SERVER_PORT: TCP port (decimal).

REQUEST_METHOD: name of HTTP method (see above).

PATH_INFO: path suffix, if appended to URL after program name and a slash.

PATH_TRANSLATED: corresponding full path as supposed by server, if **PATH_INFO** is present.

SCRIPT_NAME: relative path to the program, like `/cgi-bin/script.cgi`.

QUERY_STRING: the part of URL after `?` character. The query string may be composed of `*name=value` pairs separated with ampersands (such as `var1=val1&var2=val2...`) when used to submit form data transferred via GET method as defined by HTML application/x-www-form-urlencoded.

REMOTE_HOST: host name of the client, unset if server did not perform such lookup.

REMOTE_ADDR: IP address of the client (dot-decimal).

AUTH_TYPE: identification type, if applicable.

REMOTE_USER used for certain **AUTH_TYPE**s.

REMOTE_IDENT: see `ident`, only if server performed such lookup.

Intermediate Nodes

CONTENT_TYPE: Internet media type of input data if PUT or POST method are used, as provided via HTTP header.

CONTENT_LENGTH: similarly, size of input data (decimal, in octets) if provided via HTTP header.

Variables passed by user agent (

HTTP_ACCEPT,

HTTP_ACCEPT_LANGUAGE,

HTTP_USER_AGENT,

HTTP_COOKIE and possibly others)

contain values of corresponding HTTP headers and therefore have the same sense.

Making HTML pages by C++

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

Html.cpp
Html.hpp

welcomeTime.cpp

welcomeTime.o: Html.h

welcome: Html.o welcomeTime.o
\$(CC) Html.o welcomeTime.o -o welcome

runwelcome: welcome
./welcome

main()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.softpower.com/pub_bk05.html

```
#include "Html.h"
#include <time.h>

int main()
{
    static time_t tod;
    Html page;
    page.lead((char *) "Time to Welcome");
    page.send((char *) "<center><font size=+2><strong>");
    page.send((char *) "Welcome" );
    page.send((char *) "</strong></font></center>");
    page.send((char *) "<p>");
    page.send((char *) "We appreciate your visit. <p> Our local time is
<p>");
    page.send((char *) "<em>");
    time(&tod);
    page.send(ctime(&tod));
    page.send((char *) "</em>");
    page.trailer();
    return 0;
}
```

lead()

send()

trailer()

Content-Type: text/HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
<HTML><HEAD><rTITLE>Time to Welcome</TITLE>
</HEAD><BODY>
```

```
<center><font size=+2><strong>
Welcome
</strong></font></center>
<p>
We appreciate your visit.
<p>
Our local time is
<p>
<em>
Tue Feb 17 10:50:18 2015
</em>
```

```
</BODY>
</HTML>
```

leader() and trailer()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
//////// Html.C //////////
#include <fstream>
#include "Html.h"
using std::ifstream;

void Html::leader(char* title)
{ if ( started ) return;

  cout << "Content-Type: text/HTML" << "\r\n\r\n"
        << version
        << "<HTML><HEAD><rTITLE>"
        << title << "</TITLE>\n";

  if ( bodyOpen==NULL ) cout << "</HEAD><BODY>\n";
  else sendFile(bodyOpen);

  started = true;
}

void Html::trailer()
{ if ( bodyClose==NULL ) cout << "</BODY>\n";
  else sendFile(bodyClose);

  cout << "</HTML>\n";

  started = false;
}
```

In main()c

```
page.leader((char *) "Time to Welcome");
```

Content-Type: text/HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
<HTML><HEAD><rTITLE>Time to Welcome</TITLE>
</HEAD><BODY>
```

```
</BODY>
</HTML>
```

Class Html Constructor

```
Html(char* f1=NULL, char* f2=NULL)
: bodyOpen(f1), bodyClose(f2),
  version((char *) "<!DOCTYPE HTML PUBLIC "
           "\"-//W3C//DTD HTML 3.2 //EN\">\n")
{ started = false; }
```

P.S. Wang, "Standard C++ with objected-oriented programming"

sendFile() and send()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

Read "file" and write it into standard output

```
void Html::sendFile(char* file)
{ if ( !started ) leader((char *) "No Title");

  ifstream in(file);
  char c[1];

  while ( in.read(c, 1) ) cout.write(c,1);
}

void Html::send(char *s)
{ if ( ! started ) leader((char *) "No Title");

  cout << s;
}
```

```
//////// Html.C //////////
#include <fstream>
#include "Html.h"
using std::ifstream;

void Html::leader(char* title)
{ if ( started ) return;

  cout << "Content-Type: text/HTML" << "\r\n\r\n"
    << version
    << "<HTML><HEAD><rTITLE>"
    << title << "</TITLE>\n";

  if ( bodyOpen==NULL ) cout << "</HEAD><BODY>\n";
  else sendFile(bodyOpen);

  started = true;
}

void Html::trailer()
{ if ( bodyClose==NULL ) cout << "</BODY>\n";
  else sendFile(bodyClose);

  cout << "</HTML>\n";

  started = false;
}
```

Class Html header file

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.softpower.com/pub_bk05.html

```
#ifndef __Html_SEEN__
#define __Html_SEEN__
//////// Html.h //////////
#include <iostream>
using std::cout;

class Html
{ public:
    Html(char* f1=NULL, char* f2=NULL)
      : bodyOpen(f1), bodyClose(f2),
        version((char *) "<!DOCTYPE HTML PUBLIC "
                 "\"-//W3C//DTD HTML 3.2 //EN\">\n")
    { started = false; }

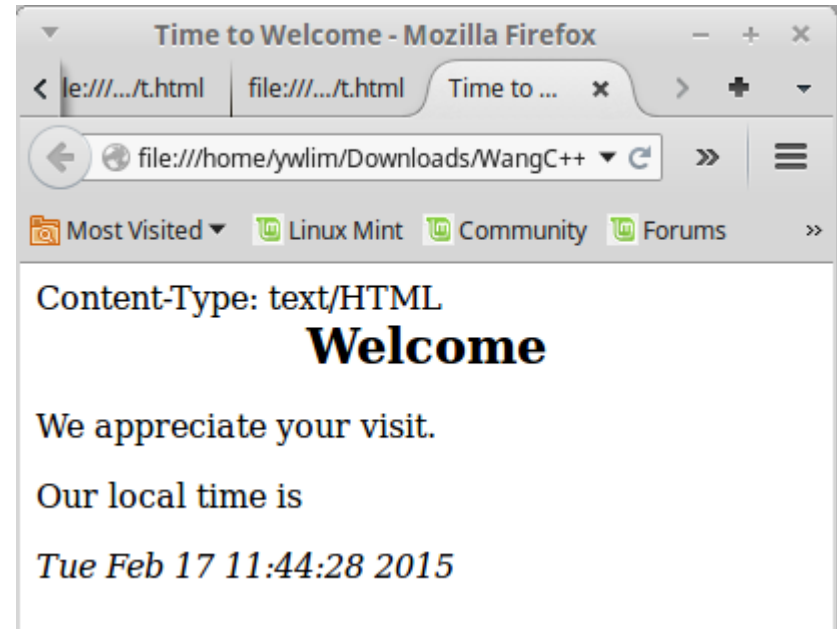
    void leader    (char* title);
    void trailer   ();
    void send      (char *s);
    void sendFile  (char* file);

    char* version;

private:
    bool  started;
    char* bodyOpen;
    char* bodyClose;
};
#endif
```

HTML File Output

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html



```
ywlim@ywlim-DeskTop-System ~/Downloads/WangC++/ex12 $ ./welcome
Content-Type: text/HTML

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
<HTML><HEAD><TITLE>Time to Welcome</TITLE>
</HEAD><BODY>
<center><font size=+2><strong>Welcome</strong></font></center><p>We appreciate y
our visit. <p> Our local time is <p><em>Tue Feb 17 11:44:22 2015
</em></BODY>
</HTML>
```


Decoding URL-encoded Data

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

Tokenizer.cpp
Tokenizer.hpp

testTokenizer.cpp

testToken1() & testToken()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.softpower.com/pub_bk05.html

```
//////// testTokenizer.C //////////
#include <iostream>
#include "Tokenizer.h"
using namespace std;

void testToken1()
{ string t(" name John Doe age 19 ");
  Tokenizer tk(t, " "); // " " //

  while ( tk.moreToken() )
  { cout << tk.tokenCount() << "=";
    cout << tk.nextToken() << endl;
  }
}

void testToken()
{ string t("name=John+Doe");
  Tokenizer tk(t, "="); // "=" //

  while ( tk.moreToken() )
  { cout << tk.tokenCount() << "=";
    cout << tk.nextToken() << endl;
  }
}
```



```
5=name
4=John
3=Doe
2=age
1=19
```



```
2=name
1=John+Doe
```

testDelTokenizer() & main()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.softpower.com/pub_bk05.html

```
void testDelToken()
{
    string t("Test:of::delimiters::also:as:tokens.::");
    string del("::"); // "::" //
    Tokenizer tk(t, del, true);

    while ( tk.moreToken() )
    {
        cout << tk.tokenCount() << "=";
        cout << tk.nextToken() << endl;
    }
}

int main()
{
    testToken1();
    testToken();
    testDelToken();
    return 0;
}
```



```
16=Test
15=:
14=of
13=:
12=;
11=delimiters
10=:
9=;
8=:
7=also
6=:
5=as
4=:
3=tokens.
2=;
1=;
```

Class Tokenizer Header

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
#ifndef __Tokenizer_SEEN__
#define __Tokenizer_SEEN__
//////// Tokenizer.h //////////
#include <string>
using std::string;

class Tokenizer
{ public:
    Tokenizer(const string& str,
              const string& delim = WHITE,
              bool want_delim = false)
: target(str), position(0), delimiter(delim),
  delimToken(want_delim) { }

    bool moreToken() const;
    string nextToken();
    string setDelimiter(const string& delim) { delimiter = delim; }
    int tokenCount() const;

private:
    string next();
    string nextAll();
    static const string WHITE;
    int position;
    const string& target;
    string delimiter;
    bool delimToken;
};
#endif
```

moreToken(), nextToken(), next()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
//////// Tokenizer.C //////////
#include "Tokenizer.h"

const string Tokenizer::WHITE=string("\r \n\t");

bool Tokenizer::moreToken() const
{   if ( position == target.length() ) return false;
    if ( delimToken ) return true;
    int i0 = target.find_first_not_of(delimiter, position);
    return(i0 != string::npos);
}

string Tokenizer::nextToken()
{   if ( delimToken ) return nextAll();
    else return next();
}

string Tokenizer::next()
{   int i0=position;
    int i1=position;           // start and end indices
    if ( i0 >= target.length() ) return string();
    i0 = target.find_first_not_of(delimiter, i1);
    if ( i0 == string::npos ) return string();
    i1 = target.find_first_of(delimiter, i0);
    if ( i1 == string::npos ) i1 = target.length();
    position = i1;
    return string(target, i0, i1-i0); // token found
}
```

string::npos → -1, the end of a string

find_first_not_of(str, pos) → search for the first character that does not match any of the characters specified in str beginning at the position pos

nextAll(), tokenCount()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
string Tokenizer::nextAll()
{
    int i0=position;
    int i1=position;           // start and end indices
    if ( i0 >= target.length() ) return string();
    if ( delimiter.find(target[i0]) != string::npos )
    {
        i1 = i0+1;
    }
    else
    {
        i1 = target.find_first_of(delimiter, i0);
        if ( i1 == string::npos ) i1 = target.length();
    }
    position = i1;
    return string(target, i0, i1-i0); // token found
}

int Tokenizer::tokenCount() const
{
    if ( position == target.length() ) return 0;
    int count = 0;
    int i0=position, i1=position;
    while ( 1 )
    {
        if ( i1 == string::npos ) return count;
        i0 = target.find_first_not_of(delimiter, i1);
        if ( i0 == string::npos )
            return (! delimToken ? count :
                count + target.length() - i1);
        count++;
        if ( delimToken ) count += i0-i1;
        i1 = target.find_first_of(delimiter, i0);
    }
}
```


Receiving URL-encoded Form Data

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

FormData.cpp
FormData.hpp
Tokenizer.cpp
Tokenizer.hpp

testFormData.cpp

FORMOBJ = FormData.o Tokenizer.o testFormData.o

testForm: \$(FORMOBJ)
\$(CC) \$(FORMOBJ) -o testForm

runForm: testForm
export REQUEST_METHOD=POST; export CONTENT_LENGTH=94; \
./testForm < line

Receiving URL-encoded Form Data

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
//////// testFormData.C //////////
#include <iostream>
#include "FormData.h"
using std::cout; using std::endl;
using std::cin; using std::getline;

int main()
{   FormData input;

    if ( ! input )
    {   input.displayError();
        return 1;
    }

    cout << "METHOD = " << input.getMethod() << endl;
    cout << "CONTENT = " << input.getContent() << endl;
    cout << "key count = " << input.keyCount() << endl;

    if ( input.hasValue("name") )    cout << input["name"] << endl;
    if ( input.hasValue("address") ) cout << input["address"] << endl;
    if ( input.hasValue("email") )   cout << input["email"] << endl;
    if ( input.hasValue("phone") )   cout << input["phone"] << endl;

    return 0;
}

// setenv REQUEST_METHOD POST
// setenv CONTENT_LENGTH 94
// a.out < line
```

```
METHOD = POST
```

```
CONTENT = name=Paul+S%2E+Wang&address=Math%2fCS%
2C+Kent+State+U%2E%2C+OH&email=pwang%
40icm.mcs.kent.edu
```

```
key count = 3
```

```
Paul S. Wang
Math/CS, Kent State U., OH
pwang@icm.mcs.kent.edu
```

Receiving URL-encoded Form Data

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
class FormData
{ public:
    FormData();
    const string& operator [] (string key);

    const string& key(int i) const
    { return (i < name.size()) ? name[i] : empty; }

    int keyCount() const { return name.size(); }

    void enter(string key, string value)
    { data[key]=value; }

    const string& getMethod() const
    { if ( error==NO_E ) return method; }

    const string& getContent() const
    { if ( error==NO_E ) return content; }

    bool hasValue(string key) const
    { return (data.find(key) != data.end()); }

    bool operator !()
    { return error != NO_E; }

    void displayError(ostream& out=cerr) const;

    enum ERROR {NO_E, METHOD_E, LENGTH_E, CONTENT_E};
```

Receiving URL-encoded Form Data

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
#ifndef __FormData_H_SEEN__
#define __FormData_H_SEEN__
////////  FormData.h  //////////
#include <iostream>
#include <string>
#include "../ex06/token/Tokenizer.h"
#include <vector>
#include <map>
using namespace std;

private:
    ERROR            inputContent();
    void             unpack();
    static void      urlDecode(string& v); // v decoded

    map<string, string> data;
    vector<string>      name;
    string              content;
    string              method;
    string              empty;
    ERROR              error;
};
#endif
```

Receiving URL-encoded Form Data

```
//////// FormData.C ////////// P.S. Wang, "Standard C++ with objected-oriented programming", http://www.softpower.com/pub_bk05.html
#include <stdlib.h>
#include "FormData.h"

FormData::FormData()
{ content = empty = method = "";
  error = inputContent();
  if ( error == NO_E ) unpack();
}

FormData::ERROR FormData::inputContent()
{ char *s = getenv("REQUEST_METHOD");
  if ( s == NULL ) return METHOD_E; // no method
  method = s;
  if (method == "POST")
  { s =getenv("CONTENT_LENGTH");
    if ( s == NULL || *s == '\0' ) return LENGTH_E; // no length

    char* t = s ;
    int len=strtol(s, &t, 10);
    if ( *t != '\0' ) return LENGTH_E; // invalid length

    char c;
    while ( len-- > 0 && cin.get(c) ) content += c;
    if (len > 0) return CONTENT_E; // missing content
  }
  else if (method == "GET")
  { s = getenv("QUERY_STRING");
    if ( s == NULL || *s == '\0' ) return CONTENT_E;
    content = s;
  }
  else return METHOD_E; // unknown method
  return NO_E;
}
```

```
int main()
{ FormData input;
```

file "line"

```
name=Paul+S%2E+Wang&
address=Math%2fCS%2C+Kent+State+U%2E%2C+OH&
email=pwang%40icm.mcs.kent.edu|
```

in makefile


```
runForm: testForm
export REQUEST_METHOD=POST; \
export CONTENT_LENGTH=94; \
./testForm < line
```

Receiving URL-encoded Form Data

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
void FormData::unpack()
{
    Tokenizer tk(content, "&");
    string s1;
    while ( tk.moreToken() )
    {
        Tokenizer nv(s1=tk.nextToken(), "=");
        int count=nv.tokenCount();
        if ( count == 2 ) // assume one or two tokens
        {
            string nm    = nv.nextToken();
            string value = nv.nextToken();
            urlDecode(value);
            data[nm]=value;
            name.push_back(nm);
        }
        else if ( count == 1 )
            data[nv.nextToken()]="";
    }
}
```

```
FormData::FormData()
{
    content =empty = method = "";
    error = inputContent();
    if ( error == NO_E ) unpack();
}
```



Receiving URL-encoded Form Data

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
void FormData::urlDecode(string& s)
{  int x1, x2, len = s.size(), pos=0;
   char c, space = ' ';
   for (int i=0; i < len; i++)
   {  c = s[i];
      if ( c == '+' ) s[pos++] = space;
      else if ( c == '%' )
      {  char x[3] = {s[++i], s[++i], '\0'};
         char* t = x;
         int hex=strtol(x, &t, 16);
         if (*t != '\0') // hex seq invalid
         {  s[pos++]=c;
            s[pos++]=x[0];
            s[pos++]=x[1];
         }
         else
            s[pos++] = static_cast<char>(hex);
      }
      else if ( pos == i ) pos++;
      else s[pos++]=c;
   }
   s.erase(pos, len-pos); // shorten string
}
```

```
void FormData::unpack()
{  Tokenizer tk(content, "&");
   string s1;
   while ( tk.moreToken() )
   {  Tokenizer nv(s1=tk.nextToken(), "=");
      int count=nv.tokenCount();
      if ( count == 2 ) // assume one or two tokens
      {  string nm    = nv.nextToken();
         string value = nv.nextToken();
         urlDecode(value);
         data[nm]=value;
         name.push_back(nm);
      }
      else if ( count == 1 )
         data[nv.nextToken()]="";
   }
}
```

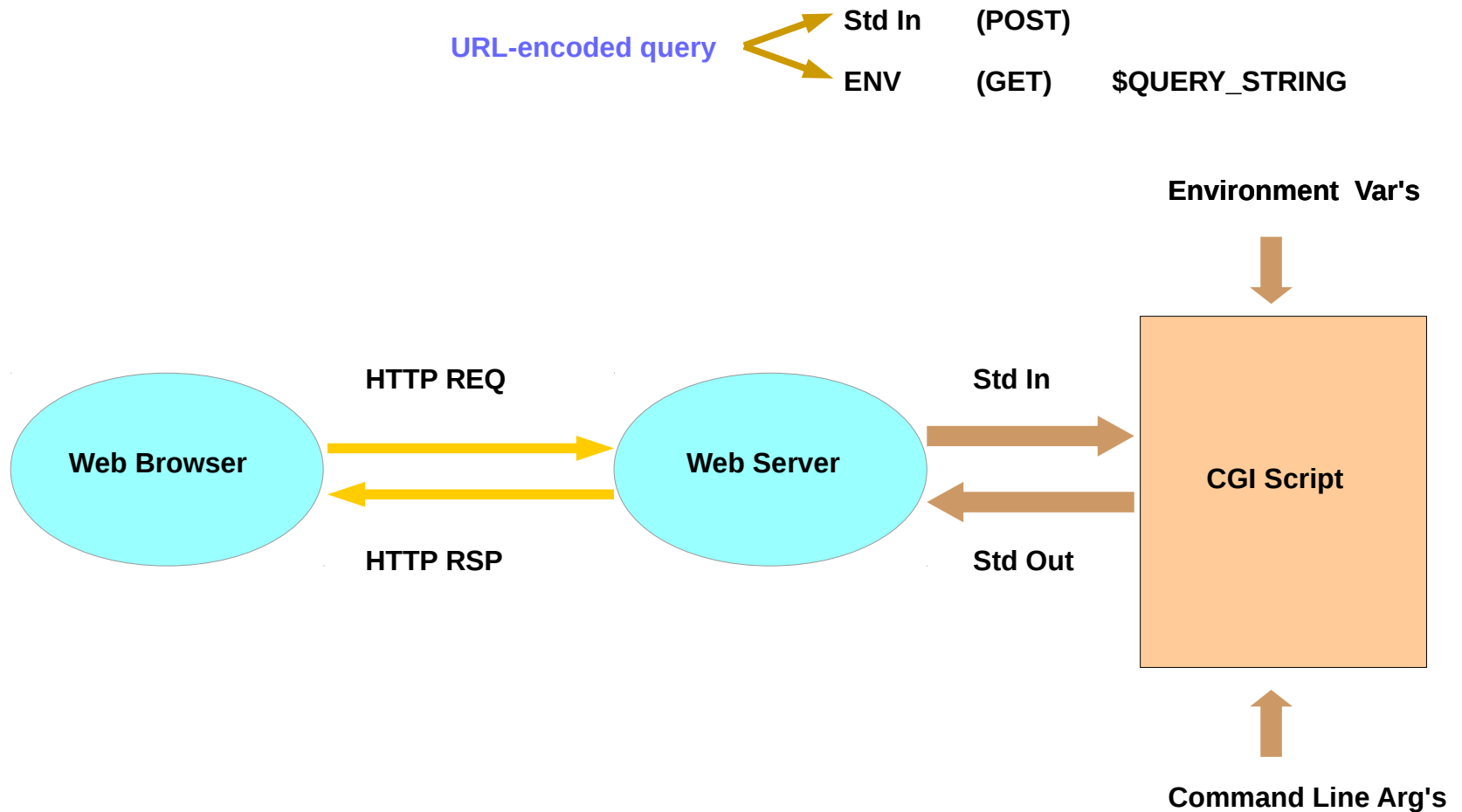
Receiving URL-encoded Form Data

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.softpower.com/pub_bk05.html

```
const string& FormData::operator [] (string key)
{   if ( error == NO_E && hasValue(key) ) return data[key];
    else return empty;
}

void FormData::displayError(ostream& out) const
{   switch(error)
    {   case METHOD_E:  out << "Invalid REQUEST_METHOD." << endl;
        break;
        case LENGTH_E: out << "Invalid CONTENT_LENGTH." << endl;
        break;
        case CONTENT_E: out << "Missing content." << endl;
        break;
    }
}
```


HTTP Request and Response



HTTP Request and Response

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

Html.cpp
Html.hpp

welcomeTime.cpp

SUGGESTOBJ = Html.o FormData.o Tokenizer.o suggest.o

```
suggest: $(SUGGESTOBJ)  
$(CC) $(SUGGESTOBJ) -o suggest
```

```
runsuggest: suggest  
export REQUEST_METHOD=POST; export CONTENT_LENGTH=253; \  
./suggest < feedback
```

Output Results

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.softpower.com/pub_bk05.html

Content-Type: text/HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
<HTML><HEAD><TITLE>Thank You</TITLE>
</HEAD><BODY>
<h3>Thank You</h3><p>Your message has been logged.<p>We
confirm receipt of the following:<p><pre>

Subject: Please state shipping charge
From: pwang@icm.mcs.kent.edu <Paul S. Wang>
Date: Fri Feb 20 19:28:37 2015

I am very interested in your home cleaning products.
But there is no shipping charge information.
It would be nice to have such info on your Web site.

</pre></BODY>
</HTML>
```

Content-Type: text/HTML

Thank You

Your message has been logged.

We confirm receipt of the following:

```
Subject: Please state shipping charge
From: pwang@icm.mcs.kent.edu
Date: Fri Feb 20 19:28:37 2015
```

I am very interested in your home cleaning products.
But there is no shipping charge information.
It would be nice to have such info on your Web site.

main()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
int main()
{
    const char    *LOGDIR = "./feedback_log/"; // dir location
    const int     SIZE = 256;
    time_t        tod;
    ostringstream mystr;

    time(&tod);
    mystr << "./feedback_log/" << tod << '\0';

    string        fname(mystr.str());
    char          *logfile = (char *) fname.c_str();
    Html          page;
    FormData      form;

    if ( ! form || ! form.hasValue("subject")
        || ! form.hasValue("email")
        || ! form.hasValue("comment")
        )
    {
        errorReply(page);
        return 1;
    }

    ofstream      ofs(logfile, ofstream::out);

    if ( ! ofs )
    {
        sorryReply(page);
        return 1;
    }

    enterLog(ofs, form, tod);
    okReply(page, logfile);

    return 0;
}
```

Input and Output Files

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

file "feedback"

```
name=Paul+S%2E+Wang&
subject=Please+state+shipping+charge&
email=pwang%40icm.mcs.kent.edu&
comment=I+am+very+interested+in+your+home+cleaning+products%2E++
But+there+is+no+shipping+charge+information%2E+
It+would+be+nice+to+have+such+info+on+your+Web+site%2E|
```

Working directory

Name	Size	Type	Date Modified
▶ feedback_log	5 items	folder	Sat 21 Feb 2015 06:45:04 PM KST
feedback	254 bytes	plain text document	Sun 22 Aug 1999 12:14:27 PM KST

Sub-directory [feedback_log](#)

Name	Size	Type	Date Modified
1009816500.txt	268 bytes	plain text document	Tue 01 Jan 2002 01:35:00 AM KST
1424510375	268 bytes	email message	Sat 21 Feb 2015 06:19:35 PM KST
1424511662	268 bytes	email message	Sat 21 Feb 2015 06:41:02 PM KST
1424511904	268 bytes	email message	Sat 21 Feb 2015 06:45:04 PM KST
t.txt	268 bytes	plain text document	Sat 21 Feb 2015 06:18:05 PM KST

enterLog() & errorReply()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.softpower.com/pub_bk05.html

```
//////// suggest.C //////////
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <time.h>
#include "Html.h"
#include "FormData.h"
using namespace std;

void enterLog(ofstream& out, FormData& form, time_t tod)
{
    out << "Subject: " << form["subject"] << endl
        << "From: " << form["email"]
        << " <" << form["name"] << ">" << endl
        << "Date: " << ctime(&tod) << endl;
    out << endl
        << form["comment"] << endl;
    out.close();
}

void errorReply(Html& page)
{
    page.leader((char *) "Form Incomplete");
    page.send((char *) "<h3>Please Complete All Information</h3><p>"
        "There are missing entries on your form.<p> "
        "Please go back, complete the form, "
        "and submit it again.<p> Thanks.<p>");
    page.trailer();
}
```

```
enterLog(out, form, tod);
okReply(page, logfile);
```

```
if ( ! form || ! form.hasValue("subject")
    || ! form.hasValue("email")
    || ! form.hasValue("comment")
    )
{
    errorReply(page);
    return 1;
}
```

sorryReply() & okReply()

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

```
void sorryReply(Html& page)
{
    page.leader((char *) "Sorry");
    page.send((char *) "<h3>Unable to process you message</h3><p>"
              "Sorry, we ran into some difficulty "
              "submitting your message. Please "
              "enter your message again at a later time.<p>");
    page.trailer();
}

void okReply(Html& page, char* file)
{
    page.leader((char *) "Thank You");
    page.send((char *) "<h3>Thank You</h3><p>"
              "Your message has been logged.<p>"
              "We confirm receipt of the following:<p><pre>\n\n");
    page.sendFile(file);
    page.send((char *) "\n</pre>");
    page.trailer();
}
```

```
if ( ! out )
{
    sorryReply(page);
    return 1;
}
```

```
enterLog(out, form, tod);
okReply(page, logfile);
```

HTTP Request and Response

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

HTTP Request and Response

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

HTTP Request and Response

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

HTTP Request and Response

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

HTTP Request and Response

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

HTTP Request and Response

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

HTTP Request and Response

<http://stackoverflow.com/questions/238547/how-do-you-programmatically-download-a-webpage-in-java>

```
public static void main(String[] args) {
    URL url;
    InputStream is = null;
    BufferedReader br;
    String line;

    try {
        url = new URL("http://stackoverflow.com/");
        is = url.openStream(); // throws an IOException
        br = new BufferedReader(new InputStreamReader(is));

        while ((line = br.readLine()) != null) {
            System.out.println(line);
        }
    } catch (MalformedURLException mue) {
        mue.printStackTrace();
    } catch (IOException ioe) {
        ioe.printStackTrace();
    } finally {
        try {
            if (is != null) is.close();
        } catch (IOException ioe) {
            // nothing to see here
        }
    }
}
```

P.S. Wang, "Standard C++ with objected-oriented programming", http://www.sofpower.com/pub_bk05.html

Reference

References

[1] <http://en.wikipedia.org/>

[2] P.S. Wang, “Standard C++ with objected-oriented programming”