Programare Web

Curs 4

Scripturi CGI in limbaje compilate

Obiectiv

- În acest capitol vom prezenta printr-un exemplu modul în care se pot scrie scripturi folosind limbajul C.
- Programul exemplifica modul in care se realizeaza in practica specificatia CGI – Common Gateway Interface

CGI

- Common Gateway Interface (CGI) este un standard pentru interfatarea aplicatiilor externe cu servere de diferse tipuri, inclusiv cu servere HTTP (servere de web)
- In mod normal, un server de web serveste la cerere un document HTML (sau de alta natura) care exista in structura sa de directoare, deci este static: un fisier stocat care este mereu acelasi.
- Un program CGI insa este executat in momentul in care apare cererea pe baza unor parametri primiti (input) si din aceasta cauza rezultatul sau (output) poate fi diferit de la o executie la alta.

CGI

- De exemplu putem scrie un program care returneaza un document continand datele existente intr-o baza de date – creat pe baza rezultatului unei cereri SQL.
- Un astfel de program este un mijlocitor (gateway) intre sistemul de gestiune de baze de date si serverul web care deserveste clientii. De aici bine si numele acestei specificatii.

CGI

- Un script (program) CGI este deci un program executabil.
- Executia scriptului la cererea clientului este echivalenta cu a permite unor terti sa execute un program in sistem, ceea ce poate avea consecinte neplacute din punct de vedere al securitatii.
- Din aceasta cauza exista o serie de restrictii in ceea ce priveste scripturile CGI.

CGI

- Una dintre cele mai comune restrictii este aceea ca ele sunt stocate intr-un director separat, cu nume predefinit la configurarea sistemului serverului de web.
- In felul acesta serverul stie ca acele fisiere trebuiesc executate si nu expediate catre client.
- In acelasi timp acest director este accesibil de obicei doar webmasterului, pentru a preintampina o serie de brese de securitate.

CGI

- Ori de cate ori o sesiune de browser trimite o cerere pentru URL-ul unui script CGI, serverul executa acel script.
- Browserul va primi ca raspuns rezzultatul (outputul) scriptului – deci ce scrie acesta la standsrd output.
- Un script CGI nu poate fi rulat (de cele mai multe ori) separat, la promptul de sistem. De exemplu, daca avem un script a.cgi, nu vom obtine rezultatele scontate tastand la promptul shell ceva de genul:

```
a.cgi –x –h date intrare
```

CGI

- Pentru transmiterea datelor de intrare, in locul liniei de comanda se folosesc variabile de mediu (environment).
- Doua variabile de mediu folosite pentru aceasta sunt:
  - QUERY_STRING
  - CONTENT_LENGHT

QUERY_STRING

- QUERY_STRING este un sir de caractere definit ca fiind ceea ce urmeaza dupa primul ? din URL-ul resursei CGI respective.
- Sirul poate proveni din datele trimise de un formular cu metoda GET sau de un ISINDEX (mai putin folosit in prezent).
- Sirul poate fi codificat si manual in URL respectand regulile de scriere
- Exemplu:

```
http://server.cs.pub.ro/cgi-bin/a.cgi?cerere=afisare&list=da&pag=1&number=25
```

QUERY_STRING

- Sirul este codificat in format standard pentru URL, deci schimband spatiile in + si codificand caracterele speciale in format hexa: %xy.
- Scriptul va trebui sa decodifice sirul respectiv inainte de folosire
- Exemplu:

```
http://server.cs.pub.ro/cgi-
    bin/a.cgi?Select+*+from+studenti
    +where+nume+like+%27a%25%27&pag=
    1
```

CONTENT_LENGHT

- In cazul in care formularul foloseste metoda POST parametrii scriptului nu mai sunt accesibili in URL si nu vor fi regasiti nici in QUERY_STRING.
- In schimb datele (in aceeasi forma) sunt puse in intrarea standard a scriptului (standard input).
- Sirul nu se sfarseste cu EOF, si din aceasta cauza se foloseste variabila de mediu CONTENT_LENGHT pentru a specifica numarul de caractere care trebuiesc citite din standard input.

Output

- Una dintre cele mai frecvente erori in scrierea scripturilor CGI este nerespectarea formatului datelor generate de acesta.
- Un script CGI poate returna un numai fisiere HTML ci orice tip de fisiere: imagini, fisiere Word, Excel, PDF, etc.
- Din aceasta cauza documentul generat este precedat de un antet care spune tipul acestuia.

Output

- Formatul este deci:

Antet

[o linie goala]

Document

- In antet trebuie sa existe una dintre directivele urmatoare:
  - Content_Type care specifica tipul documentului
  - Location care specifica o redirectare

Content-Type

- Un exemplu de document:

```
Content-type: text/html
Aici este o linie goala
<HTML>
<HEAD><TITLE>Script CGI</TITLE></HEAD>
<BODY><H1>Rezultat script CGI</H1>
Acesta este rezultatul
</BODY>
</HTML>
```

Content-Type

- Un exemplu de document:

```
Content-type: text/html
```

- Directiva din antet arata ca documentul care urmeaza este un fisier HTML
- In locul lui text/html sau text/plain pot fi o multitudine de alte elemente, ca de exemplu:
  - image/gif,
  - image/x-xbitmap,
  - image/jpeg,
  - image/pjpeg, - pentru imagini
  - application/vnd.ms-excel,
  - application/vnd.ms-powerpoint,
  - application/msword, - pentru fisiere MS Office
  - application/x-shockwave-flash, - pentru fisiere flash
  - etc

Location

- Un antet continand Location va duce la o redirectare catre URL-ul continut.
- Exemplu:

```
Location: http://server.cs.pub.ro/index.php
Aici este o linie goala
<HTML>
<HEAD><TITLE>Documentul s-a mutat</TITLE></HEAD>
<BODY> <H1>Nu mai e aici</H1>
Dar il gasesti <a href="http://ceva.cs.pub.ro/index.php
    ">aici </a>
</BODY>
</HTML>
```

Location

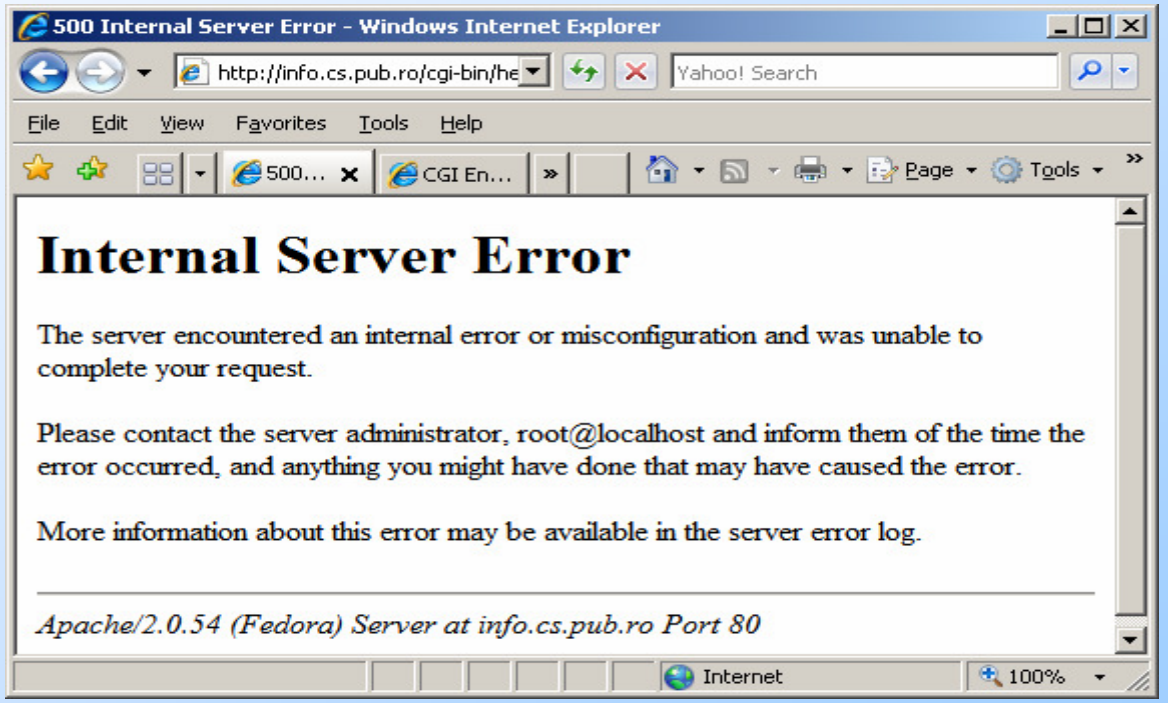
- In acest caz browserul se redirecteaza direct la noua adresa (specificata in Location).
- Continutul efectiv al paginii respective nu mai este afisat:

```
<HTML>
<HEAD><TITLE>Documentul s-a mutat</TITLE></HEAD>
<BODY> <H1>Nu mai e aici</H1>

. . . . . .
```

Erori

- In cazul in care outputul scriptului CGI nu este corespunzator se obtine o eroare de tipul “Internal server error”:



Erori

- Astfel de erori se obtin si daca plasam pagini HTML sau alte fisiere care nu sunt executabile in directorul rezervat scripturilor CGI.
- In acest caz, la cererea acestor pagini serverul incearca sa le execute in loc de a le trimite asa cum sunt catre client.
- Acest comportament se obtine si daca plasam scripturi php in directorul respectiv.

Programul C propus

- Vom considera o pagina web *cerere.html* continand un formular:

```
<html><head><title>Interogare SQL</title></head>
<body>
<form ACTION= "cgi-bin/cerere.cgi" METHOD="POST">
Cerere SQL: <input TYPE="text" NAME="cerere" SIZE=60>
<p>
<input TYPE="submit" VALUE="[Send]">
<input TYPE="reset" VALUE="[Reset]">
<hr>
Listare variabile:
<input TYPE="radio" NAME="listtot" VALUE="d">Da
<input TYPE="radio" NAME="listtot" VALUE="n" CHECKED>Nu
</form>
</body>
```

Structura program

- Etapele execuţiei scriptului sunt următoarele:

1. Trimiterea unui header corect conţinând în cazul de faţă directiva *Content-type*. Această operaţie trebuie făcută cât mai devreme pentru a nu exista posibilitatea apariţiei altui mesaj (de exemplu mesaj de eroare) în această zonă a răspunsului.
2. Colectarea argumentelor. În cazul metodei POST şirul care le conţine se găseşte la intrarea standard (*stdin*) şi are o lungime dată de variabile CONTENT_LENGTH iar în cazul metodei GET acestea sunt în variabila QUERY_STRING.
3. Setarea unor variabile ale programului cu valorile simbolilor primiţi de la formă.

Structura program

- Acestea vor conţine:
- 3.1. un sir de caractere continand o cererea generica (intr-un caz concret poate fi o cerere SQL daca scriptul afiseaza rezultatele unei astfel de cereri)
- 3.2. Opţiunea de listare a variabilelor, simbolilor primiţi de la formă şi a altor mesaje

4. Listarea simbolilor primiţi de la formă, în cazul în care s-a cerut aceasta.
5. Trimiterea unui mesaj de sfârşit şi terminarea programului.

Declaratii si variabile

- Programul a fost dezvoltat pe un sistem de tip Unix folosindu-se un stil de programare cât mai simplu pentru a permite înţelegerea sa şi de către cei care au cunoştinţe sumare privind acest limbaj.
- Variabilele şi declaraţiile globale ale programului sunt necesare pentru efectuarea operaţiilor comune oricărui CGI - decuparea din datele primite de la formă a numelor simbolilor şi a valorilor asociate cu aceştia.

Declaratii si variabile

```
#include <stdio.h>
#include <time.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>

#define         LUNGIME_SIMBOL  20
#define         LUNGIME_VALOARE 512
#define         GET     1    /* valoare transmisa cu
    GET */
#define         POST    2    /* valoare transmisa cu
    POST */
```

Declaratii si variabile

- Programul va crea o listă simplu înlănţuită de simboli.
- Fiecare element al listei conţine numele simbolului, valoarea sa, metoda prin care a fost transmis (GET sau POST) şi înlanţuirea către simbolul următor.
- S-a convenit ca numele şi lungimea unui simbol să nu fie mai mare de 20 respectiv 512 caractere.

Declaratii si variabile

- Variabilele *capLista* şi *endLista* conţin începutul, respectiv sfârşitul acestei liste.
- În plus, programul mai foloseşte variabilele:
  - char *cerere;
  - int listtot;
- în care se vor prelua din lista de simboli cererea (SQL sau de alta natura) şi opţiunea de listări extinse (da=1, nu=0).

Declaratii si variabile

```
typedef struct _simbol {
        char    nume[LUNGIME_SIMBOL],
                valoare[LUNGIME_VALOARE];
        int     metoda;
        struct _simbol *next;
} simbol_t;

simbol_t *capLista = NULL,
         *endLista = NULL;
char   *cerere;
int  listtot;
```

Inceput & Sfarsit

- Funcţiile *Inceput* şi *Sfarşit* scriu la ieşirea standard header-ul răspunsului, respectiv un mesaj de sfârşit al acestuia.
- Dacă cea de-a doua funcţie nu este neapărat necesară (poate lipsi complet), în ceea ce priveşte funcţia *Inceput* acţiunea efectuată de aceasta este esenţială.

Inceput & Sfarsit

- Ea trebuie să trimită un header care conţine cel puţin directiva *Content-type* şi care este separat de conţinutul documentului de o linie goală.
- În cazul exemplului prezentat aceste funcţii conţin şi directivele HTML care sunt specifice începutului şi sfârşitului unui document de acest tip.

Inceput

```
void Inceput()
{
   printf("Status: 200 OK\n");
   printf("Content-type: text/html\n\n");

  printf("<html><head><title>Rezultat
    cerere</title></head>\n");
  printf("<body>\n");
}
```

Sfarsit

```
void Sfarsit()
{
        printf("<hr>\n");
        printf("Copyright (c) 2000
   Autorul\n");
        printf("</body></html>\n");
        fflush(stdout);

}
```

Lista de simboli

- Lista simbolilor transmişi de formular este creată prin intermediul funcţiilor *Argumente*, *IaArgumente* şi *nouSimbol*.
- În cazul în care în şirul de perechi *simbol*=*valoare* apar caractere speciale, browserul le codifică în forma *%xy* unde *x* şi *y* sunt cele două cifre ale reprezentării hexazecimale pentru caracterul respectiv.

Lista de simboli

- În cazul în care apar astfel de situaţii scriptul trebuie să decodifice caracterul (să obţină codul său ascii).
- Această operaţie este făcută prin funcţiile *Decodifica* şi *dinHex*:
- Funcţia *dinHex* converteşte un şir de două caractere care reprezintă un cod hexazecimal în codul ascii corespunzător (număr întreg)

Lista de simboli

- Funcţia *Decodifica* primeşte ca argument un şir de caractere şi înlocuieşte toate apariţiile de forma *%xy* cu caracterul ascii decodificat prin *dinHex*.
- Sursa acestor funcţii a fost adaptată după codul NCSA.
- După construcţia listei de simboli, funcţia *SeteazaVar* caută în aceasta doi simboli cu nume predefinit:

Lista de simboli

- **cerere**: textul cererii (SQL sau de alta natura)
- **listtot**: listări extinse. Această variabilă poate avea valorile **d** sau **n.**

depunând valorile lor în variabilele globale cu acelaşi nume.

```
static char din_hex (c)
       char    c;
{
  return  c >= '0' && c <= '9' ?  c - '0'
          : c >= 'A' && c <= 'F'? c - 'A' + 10
          : c - 'a' + 10;
/* accepta litere mici */
}
```

Intrebare: de ce + 10 cand sunt cifre hexazecimale?

Decodifica

```
char * Decodifica (sir)
       char    *sir;
{   char * p = sir;
    char * q = sir;
    while(*p) {
        if (*p == '%') {
            p++;
            if (*p) *q = din_hex(*p++) * 16;
            if (*p) *q = (*q + din_hex(*p++));
            q++;
        } else {
            if (*p == '+') {
                *q++ = ' ';
                p++;
            } else {
                *q++ = *p++;

    }   }     }
    *q++ = 0;
    return sir;
} /* Decodifica */
```

Argumente

```
void Argumente()
{       char    *cerere, *metoda;
        int     lungime, rest;
       cerere = (char *)getenv("QUERY_STRING");
        if (cerere)
        { if (strlen(cerere) > 0)
           { IaArgumente(cerere, GET); } }
         metoda=(char *)getenv("REQUEST_METHOD");
        if (metoda)
        {       if (strcmp(metoda,"POST" )==0)
          {  lungime = atoi((char *)getenv("CONTENT_LENGTH"));
             cerere = (char *) malloc(lungime + 1);
             rest = lungime;
             while(rest)
             { rest -= read(fileno(stdin) , cerere + lungime - rest, rest); }
              cerere[lungime]='\0';
             IaArgumente(cerere, POST);
        } }
}
```

Identifica sirul de simboli si valori si apeleaza IaArgumente pentru a separa in perechi si a construi lista de simboli

IaArgumente

```
void IaArgumente(cerere, ceMetoda)
        char    *cerere; int     ceMetoda;

{       char    *cp1, *cp2, var[50], val[15 * 1024];
        simbol_t *simbol;

        if (!cerere) return;
        cp1 = cerere;  cp2 = var;
        bzero(var,sizeof(var)); bzero(val,sizeof(val));
        while(*cp1)
        {       if (*cp1 == '=')  /* deci in var avem acum numele variabilei */
                {       cp1++;
                        cp2 = val; /* deci de acum in val luam valoarea */
                        continue;
                }
                if (*cp1 == '&')   /* s-a terminat o definitie */
                {
                        if (strlen(val) > LUNGIME_VALOARE)
                        {       printf("Eroare: Valoare prea mare<br>\n");
                                exit(1); }
                        simbol = nouSimbol(var, ceMetoda);
                        strcpy(simbol->valoare, (char *)strdup(Decodifica(val)) );
```

IaArgumente

```
/* anulam var si val pentru urmatoarea variabila */
                        bzero(var,sizeof(var));
                        bzero(val,sizeof(val));
                        cp1++;/* trecem peste & */
                        /* incepe un nou nume de variabila */

                        cp2 = var;
                        continue;
                }
                *cp2++ = *cp1++;  /* caracter pe care-l adaugam in var sau val
*/
        }
        /* pentru ultma variabila */
        if (strlen(val) > LUNGIME_VALOARE)
        {
                printf("Eroare: Valoare prea mare<br>\n");
                exit(1);
        }

        simbol = nouSimbol(var, ceMetoda);
        strcpy(simbol->valoare, (char *)strdup(Decodifica(val)));

}
```

nouSimbol

```
simbol_t *nouSimbol(nume, ceMetoda)
        char    *nume;
        int     ceMetoda;
{
        simbol_t   *nou;

nou = (simbol_t *)malloc(sizeof(simbol_t)); /* se aloca spatiu */
        if (!nou)
        {
                printf("Eroare: Memorie insuficienta <br>\n");
                exit(1);
        }
        strcpy(nou->nume,nume);
        nou->metoda = ceMetoda;
        nou->next = NULL;
        if (!capLista)
        {
                capLista = nou;
        }
        else
        {
                endLista->next = nou;
        }
        endLista = nou;
        return(nou);
}
```

SeteazaVar

- Cauta anumiti simboli in lista si seteaza variabilele corespunzatoare din program:

```
void SeteazaVar(cap)
simbol_t *cap;
{

simbol_t *s;
s = cap;
while (s)
{
/* in functie de numele simbolului se pot efectua diverse
     setari */
if (!strcmp("cerere", s->nume))
  cerere = s->valoare;
if (!strcmp("listtot", s->nume))
        listtot = !strcmp(s->valoare, "d");
s = s->next;
} /* while */
}
```

Variabile de mediu

- În cazul opţiunii de listare extinsă, pentru tipărirea valorilor variabilelor de mediu se foloseşte o funcţie numită *oVariabila* care extrage şi tipăreşte valoarea unei variabile sau un mesaj în cazul în care variabila nu există (nu a fost setată de serverul de web).

oVariabila

```
void oVariabila(numeVariabila)
char    *numeVariabila;
{
char    *sir;

sir = (char *)getenv(numeVariabila);
if (sir)
        printf("%s = %s<br>\n",
    numeVariabila, sir);
else
    printf("%s = ***NU ESTE SETATA***<br>\n",
    numeVariabila);
}
```

Variabile de mediu

- Apelul repetat al acestei funcţii (oVariabila) duce la tipărirea tuturor variabilelor de mediu specificate în funcţia *variabile*.
- Aceste variabile sunt detaliate si in specificatia CGI.
- Lista lor este urmatoarea:

Variabile de mediu

- Exista 3 variabile de mediu care nu sunt legate de cererea in sine si deci sunt setate pentru orice tip de cerere:
- SERVER_SOFTWARE: numele si versiunea serverului care serveste cererea respectiva - si care a lansat in executie scriptul curent. Formatul este: nume/versiune.

Exemplu: Apache/2.0.54 (Fedora)

- SERVER_NAME: numele serverului (hostname, DNS alias, sau adresa IP address, asa cum apare ea in URL-urile de autoreferire)

Exemplu: www.cs.pub.ro

- GATEWAY_INTERFACE: versiunea specificatiei CGI folosita de serverul respectiv in forma CGI/revision

Exemplu: CGI/1.1

Variabile de mediu

- Variabilele urmatoare sunt specifice cererii (request) care se proceseaza:
- SERVER_PROTOCOL: numele si versiunea protocolului cu care a sosit cererea in formatul protocol/versiune.

Exemplu: HTTP/1.1

- SERVER_PORT: numarul portului catre care a fost trimisa cererea.

Exemplu: 80

- REQUEST_METHOD: Metoda folosita pentru cererea curenta. Pentru HTTP aceasta poate fi GET, HEAD sau POST

Variabile de mediu

- PATH_INFO: informatii suplimentare de cale furnizate de client. Aceste informatii sunt puse in URL dupa numele scriptului si se regasesc in PATH_INFO.

Exemplu: daca scriptul a fost apelat astfel:

```
http://info.cs.pub.ro/cgi-
   bin/cerere.cgi/mai/mult/fruct/mai/multa/energie?listtot
   =d
```

Atunci vom obtine PATH_INFO = /mai/mult/fruct/mai/multa/energie

- PATH_TRANSLATED: translatarea lui PATH_INFO in termeni absoluti (ea este relativa la radacina arborelui de documente).

Exemplu: pentru cazul anterior, daca DOCUMENT_ROOT este /usr/var/htdocs/

atunci

PATH_TRANSLATED = /usr/var/htdocs/mai/mult/fruct/mai/multa/energie

Variabile de mediu

- SCRIPT_NAME: calea relativa a scriptului care se executa.

Exemplu: SCRIPT_NAME = /cgi-bin/cerere.cgi

- QUERY_STRING: Valoarea variabilei, descrisa anterior. In cazul unui formular si a metodei POST nu este setata.
- REMOTE_HOST: numele hostului care face cererea. Daca nu exista aceasta informatie poate fi nesetata dar variabila REMOTE_ADDR poate contine atunci adresa IP a clientului.
- REMOTE_ADDR: vezi mai sus.

Exemplu: REMOTE_ADDR = 141.85.37.1

Variabile de mediu

- AUTH_TYPE: daca serverul are suport pentru autentificare user atunci aceasta variabila poate contine valoarea tipica "Basic", altfel e nula.
- REMOTE_USER: in cazul "Basic", in aceasta variabila contine un userID, altfel e nula
- CONTENT_TYPE: In cazul metodei POST specifica tipul informatiei trimise de client. CONTENT_LENGTH va contine in acest caz lungimea sirului de date trimis de client

Exemplu:

pentru sirul de date: cerere=select+*&listtot=d

Obtinem:

CONTENT_TYPE = application/x-www-form-urlencoded
CONTENT_LENGTH = 25

Variabile de mediu

Alte variabile de mediu:

- HTTP_ACCEPT: lista formatelor acceptate de client in format: tip/subtip, tip/subtip, ….
- HTTP_USER_AGENT: informatii despre browser in formatul: software/versiune biblioteca/versiune.

Exemplu:

```
HTTP_ACCEPT = image/gif, image/x-xbitmap,
   image/jpeg, image/pjpeg, application/x-
   shockwave-flash, application/vnd.ms-excel,
   application/vnd.ms-powerpoint,
   application/msword, */*

HTTP_USER_AGENT = Mozilla/4.0 (compatible; MSIE
   7.0; Windows NT 5.1; .NET CLR 2.0.50727;
   InfoPath.2; .NET CLR 1.1.4322)
```

Variabile de mediu

- De asemenea serverul va converti liniile de header ale cererii clientului in variabile cu numele:

HTTP_header

In care caracterul - este inlocuit cu _.

- Exemple de variabile de acest tip:

HTTP_HOST, HTTP_IF_MODIFIED_SINCE

Functia variabile()

```
void variabile()
{
oVariabila("SERVER_SOFTWARE");
oVariabila("SERVER_NAME");
oVariabila("GATEWAY_INTERFACE");
oVariabila("SERVER_PROTOCOL");
oVariabila("SERVER_PORT");
oVariabila("REQUEST_METHOD");
oVariabila("PATH_INFO");
oVariabila("PATH_TRANSLATED");
oVariabila("SCRIPT_NAME");
oVariabila("QUERY_STRING");
oVariabila("REMOTE_HOST");
oVariabila("REMOTE_ADDR");
oVariabila("AUTH_TYPE");
oVariabila("REMOTE_USER");
oVariabila("REMOTE_IDENT");
oVariabila("CONTENT_TYPE");
oVariabila("CONTENT_LENGTH");
oVariabila("HTTP_ACCEPT");
oVariabila("HTTP_USER_AGENT");
}
```

Programul principal

- Programul principal nu face decât să apeleze funcţiile descrise anterior, şi anume:
- Trimiterea unui header corect (cu funcţia *Inceput*)
- Crearea listei de simboli şi valori ale acestora (apelul funcţiei *Argumente*)
- Setarea variabilelor *cerere* şi *listtot* din simbolii corespunzători aflaţi în listă

Programul principal

- În cazul listărilor extinse (*listtot = TRUE*) este tiparită lista de simboli şi prin apelul funcţiei *variabile* sunt listate şi variabilele de mediu cu valorile lor.
- Trimiterea unui mesaj de sfârşit şi terminarea programului (funcţia *Sfarsit*)
- Textul programului principal este următorul:

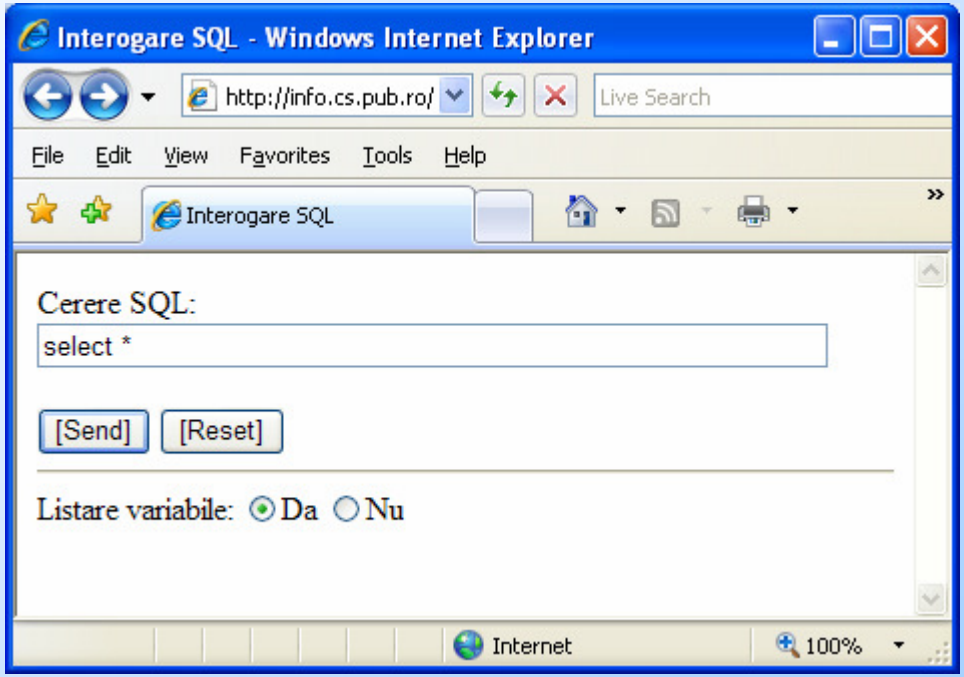
Programul principal

```
int main()
{
simbol_t *s = NULL;
Inceput();
Argumente();
SeteazaVar(capLista);

if (listtot)
{ printf("<hr><h2>Simboli primiti:</h2>\n");
  s = capLista;
  while (s)   { printf("Simbol=%s Valoare=%s ", s->nume, s->valoare);
                if (s->metoda)
                        printf("Metoda=GET<br>\n");
                else
                        printf("Metoda=POST<br>\n");
                s = s->next; }
  printf("<hr><h2>Valorile unora dintre variabilele de mediu (env)
      sunt:</h2>\n");
  variabile();
}
Sfarsit();
exit(0);
}
```

Executie

- Pagina HTML continand formularul:



Executie


Rezultat cerere - Windows Internet Explorer
http://info.cs.pub.ro/cgi-bin/cerere.cgi
Live Search
File Edit View Favorites Tools Help
Rezultat cerere
Page Tools
Simboli primiti:
Simbol=cerere Valoare=select * Metoda=POST
Simbol=listtot Valoare=d Metoda=POST
Valorile unora dintre de mediu (env) sunt:
SERVER_SOFTWARE = Apache/2.0.54 (Fedora)
SERVER_NAME = info.cs.pub.ro
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.1
SERVER_PORT = 80
REQUEST_METHOD = POST
PATH_INFO = ***NU ESTE SETATA***
PATH_TRANSLATED = ***NU ESTE SETATA***
SCRIPT_NAME = /cgi-bin/cerere.cgi
QUERY_STRING =
REMOTE_HOST = ***NU ESTE SETATA***
REMOTE_ADDR = 141.85.169.122
AUTH_TYPE = ***NU ESTE SETATA***
REMOTE_USER = ***NU ESTE SETATA***
REMOTE_IDENT = ***NU ESTE SETATA***
CONTENT_TYPE = application/x-www-form-urlencoded
CONTENT_LENGTH = 25
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
HTTP_USER_AGENT = Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 2.0.50727; InfoPath.2; .NET CLR 1.1.4322)
Copyright (c) 2000 Autorul
Done
Internet
100%

Alte setari

- Pentru executia scripturilor CGI serverul de web trebuie configurat corespunzator.
- In continuare prezentam succint unele elemente de configurare
- Elementele de configurare descrise se hasesc in fisierul de configurare al serverului, httpd.conf

ScriptAlias

- Asa cum am spus, scripturile cgi trebuiesc puse intr-un director anume.
- Acesta este definit in fisierul de configurare cu ScriptAlias
- Apache va considera toate fisierele din acest director este executabil si nu trebuie trimis la client daca este cerut ci executat.
- Exemplu:

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
```

ScriptAlias

- Asta inseamna ca la un URL de tipul:

`http://server.ro/cgi-bin/cerere.cgi`

se va executa scriptul:

`/usr/local/apache2/cgi-bin/cerere.cgi`

aflat pe serverul server.ro

User CGI

- Daca se permite utilizatorilor sa ruleze scripturi CGI din pagina lor personala (care nu se gaseste in zona de documente ale serverului ci in contul privat al userului) atunci se pot folosi AdHandler/SetHandler si Option in Directory.

User CGI

- Exemplul 1: scripturile CGI au extensia .cgi si sunt in pagina de web a userului. Elementele de configurare sunt:

```
<Directory /home/*/public_html>
Options +ExecCGI
AddHandler cgi-script .cgi
</Directory>
```

USER CGI

- Exemplul 2: scripturile CGI se gasesc in pagina din contul userului, intr-un subdirector predefinit. Elementele de configurare sunt:

```
<Directory /home/*/public_html/cgi-bin>
Options ExecCGI
SetHandler cgi-script
</Directory>
```

Drepturi

- Ca si in cazul in care se servesc pagini aflate in contul userului, trebuie ca serverul de web sa aiba acces pe calea pana la scriptul CGI (drept de execute pe directoare) pe calea catre script) si sa-l poata executa (drept de execute pe script) .
- In caz contrar vom obtine mesaje de eroare de tip Internal Server Error.

Bibliografie

- The Common Gateway Interface
  http://hoohoo.ncsa.uiuc.edu/cgi/
- Apache Tutorial: Dynamic Content with CGI
  http://httpd.apache.org/docs/2.2/howto/cgi.html