

Lucrarea 4 HTTP și CGI

1. NOTIUNI INTRODUCTIVE

CGI (Common Gateway Interface) reprezintă o interfață (un set de norme) care fac posibilă comunicarea între un server de HTTP și un program care rulează pe același calculator. Prin CGI se poate apela programul de către server, și i se pot transmite parametri.

La ce se poate folosi CGI ? De exemplu, la:

- citirea și interpretarea datelor introduse de utilizator într-un formular HTML;
- accesarea unei baze de date folosind un formular HTML
- crearea de documente în mod dinamic, la cerere;
- rularea unui program specificat de pe server, fara ca utilizatorul să dispuna de un cont (shell) pe server
- În ce limbaj se poate scrie un script CGI? În principiu, în orice limbaj capabil să citească “intrarea standard” (stdin) sau variabilele de mediu (environment) și să scrie la “iesirea standard” (stdout).

NOTA1 : Se observă deci că termenul de “script” CGI este forțat, doar shell-scripturile fiind literal scripturi, restul fiind programe compilate sau interpretate.

NOTA2 : Exista și alte moduri de a genera conținut dinamic într-o pagină de web. CGI se folosește mai ales atunci când se dorește interacțiunea cu un program extern.

Reamintim semnificația conceptelor de *stdin*, *stdout* și *environment*.

- “*stdin*” și “*stdout*”, prescurtări de la “standard input” și “standard output” reprezintă “sursa” din care programul citește datele de intrare, respectiv “destinația” la care scrie datele de ieșire. În mod obișnuit, pentru un program lucrând în mod text (la consola DOS/Windows sau UNIX), *stdin* este tastatura, iar *stdout* este monitorul. Acestea sînt “redirectionate” automat prin TCP/IP atunci când nu se lucrează direct de la consola ci de exemplu prin conectarea cu ssh. Dacă se dorește înlocuirea *stdout* cu altceva, de exemplu o comandă „ls” se dorește să scrie într-un fișier și nu pe ecran, se folosește operatorul de indirectare notat “>”, cu sintaxa:

```
ls > fisier           scrie rezultatul în fișier
ls /bin > fisier      scrie rezultatul în fișier
ls /bin | cat > fisier același lucru „mai complicat”; semnul | se citește „pipe” (conductă);
stdout-ul comenzii ls este automat vărsat în stdin-ul comenzii cat, iar acesta concatenează (copiază) în fișier
```

```
ls /bin | sort        ieșirea lui ls cu pipe la intrarea lui sort deci va fi sortată
cat /asi/cgi-bin/test-cgi | tail -2 afișează fișierul dat, dar doar ultimele 2 linii (efectul
lui tail)
```

La fel pentru citirea din altă parte decît tastatura, se folosește indirectarea dintr-un fișier folosind “<”.

- “*environment*” (mediu/variabile de mediu) se referă la rezultatul comenzii “SET” din DOS/Windows sau “setenv” și “export” din UNIX. De exemplu în DOS și UNIX exista o variabilă de mediu numită PATH, care se inițializează cu comanda:

DOS/Windows	UNIX
SET PATH=C:\DOS;C:\WINDOWS ...	export PATH=/bin:/usr/bin:/usr/local/bin

Rezultatul acestor comenzi este că sistemul de operare va memora aceste variabile, iar programele vor putea apela funcții de sistem (în C funcția este “*getenv()*”) care să întoarcă valoarea lor, spre a le folosi (astfel sistemul de operare știe unde să caute executabilele mai importante, programele știu unde să-și pună fișierele temporare, etc).

Alte comenzi care permit lucrul cu variabile de mediu:

DOS/Windows	UNIX	descriere
env (doar în 2000/XP)	env	vizualizează toate variabilele de mediu (<i>environment</i>)
path	echo \$PATH sau env grep PATH	vizualizează conținutul variabilei PATH

Am revizuit aceste noțiuni pentru că ele sînt folosite de către scripturile CGI.

APLICAȚIE A1

Fiecare user vizualizează, respectiv modifică environmentul său, folosind *env* și *export*

- creați folosind *export* o variabilă de mediu cu o valoare la întîmplare, și afișați-i valoarea. De obicei variabilele se denumesc cu litere mari. Atenție, nu lăsați spațiu în stînga și dreapta semnului egal!

Variabilele de environment sînt specifice unui anumit proces și deci unui anumit user. Se experimentează comanda “ps” care afișează procesele. Pt a vedea toate procesele folosiți „ps aux”. Se observă arborele proceselor, pornind de la procesul *init*, “străbunicul” tuturor proces. De notat că *fiecare* proces are propriul environment, moștenit de la environment-ul părintelui. Deci, un proces părinte poate transmite un environment comun la mai mulți fii, dar fiii între ei nu-și pot modifica environment-ul!

Consecință: serverul web este un proces, și procesele pe care le crează el vor moșteni environmentul creat de acesta !

Se experimentează comanda *grep* sub UNIX (*man grep*). *Grep (Global Regular Expressions Parser)* folosește expresii regulate (*regex*) ca pattern-uri de căutare. De exemplu:

```
echo acesta este un test | grep acesta      va întoarce tot rîndul căci conține token-ul
„acesta”
echo acesta este un test | grep bla          nu întoarce căci nu conține
token-ul „bla”
echo acesta este un test | grep ^acesta     ^ = la început de șir
echo acesta este un test | grep ^test      test nu e la început de șir
```

APLICAȚIE A2

- Ce comandă (combinație de ps și grep) veți da pentru a căuta în environment și afișa doar numele userului curent?

- Ce comandă veți da pentru a afișa doar procesele care aparțin userului curent ? observați că nu este suficient să puneți doar numele userului în grep, căci acest nume poate apărea și pentru procese care nu-i aparțin acestuia.

2. VARIABILE DE MEDIU SPECIFICE CGI

În momentul în care serverul lansează în execuție un script CGI, el setează un număr de variabile de mediu (environment) pe care scriptul le poate examina. Uneori scriptul preia toată informația de care are nevoie din variabilele de mediu, altele din intrarea standard sau linia de comandă, sau altele combinat.

Important! din motive de securitate, scripturile CGI nu sînt rulate de către server de oriunde, ci dintr-un director care în bara de adrese apare ca <http://www.site.com/cgi-bin> dar se află fizic nu în rădăcină (“/”), ci într-o locație configurată pe server. Rezultă că, fără permisiuni administrative, studenții nu pot accesa (și în special modifica) scripturile CGI. Pt acest laborator, o copie a scripturilor se află pe matrix în /asi/cgi-bin. În ceea ce urmează, de fiecare dată cînd vreți să examinați/rulați un script, căutați-l în /asi/cgi-bin și rulați-l de acolo (sau copiați-l în home-ul vostru).

Pagina cu aplicațiile folosite în continuare este <http://matrix.elcom.pub.ro/asi/tst.html>

APLICAȚIE A3

- 1) Se vizualizează variabilele de mediu în aplicația “afișază environment” din pagina de test.
- 2) În terminalul ssh pe matrix (mașina unde rulează serverul de web) copiați scripturile din /asi/cgi-bin în home-directory-ul echipei voastre și se rulează scriptul:

```
./test-cgi
```

De ce variabilele au valori diferite în browser (care afișează pagina de pe matrix) și în terminalul (*tot pe matrix*), din moment ce este vorba de același calculator – matrix ?

Se încearcă setarea valorilor dorite cu export variabila=valoare, de exemplu:

```
export QUERY_STRING=ABC
```

în linia de comandă, înainte de a rula scriptul în terminal, și se verifică efectul. Ce s-a schimbat?

Dintre variabilele de mediu ne interesează în special:

- QUERY_STRING

Conține efectiv parametrii cu care a fost “chemat” scriptul, sub forma unui șir cu parametrii separati prin semnul “&”. Acesta este șirul care a fost “pasat” scriptului după semnul “?”. Tipic, șirul este de forma *nume1=val1&nume2=val2&nume3=val3...*

- REQUEST_METHOD

Metoda prin care a venit cererea. Pentru HTTP, metoda poate fi GET, POST sau HEAD. În formularul din pagina de test, se poate vedea că s-a specificat în mod explicit METHOD=”POST”.

- CONTENT_TYPE

Pentru cererile care au atașată o informație, cum ar fi HTTP POST, aceasta variabila specifică tipul datelor.

- CONTENT_LENGTH

Conține lungimea datelor trimise de client; astfel, browserul va ști câte caractere să citească din *query-string*.

APLICAȚIE A4

Cum puteți modifica valoarea variabilei `QUERY_STRING` din environmentul afișat în browser (în pagina de test) ? dar cea din terminal ? Ce *metodă* este folosită în câmpul `REQUEST_METHOD` ? Țineți minte această metodă, va fi folosită și ulterior, la pasarea parametrilor prin URL.

3. PRODUCEREA DE INFORMAȚII (output) DE CĂTRE UN CGI

Rareori un script CGI este “mut”, de obicei el produce date sub o anumită formă. Cel mai uzual este ca el să genereze sintaxa HTML, pe care serverul le va trimite browserului.

O greșeală frecventă este generarea directă de către script a unui document HTML, ca și când acest document ar fi scris de mână și plasat într-un folder al serverului. De fapt, pentru ca serverul să înțeleagă despre ce tip de document este vorba, trebuie generat mai întâi un header (pentru că nu orice script produce un document HTML, altele produc de exemplu grafice).

Headerul poate fi de una din formele:

1) `Content_type: tip/subtip \n\n`

“tip” și “subtip” sînt tipuri standard, iar `\n\n` semnifică lasarea unei linii goale după header. De exemplu pentru un document HTML headerul trebuie să fie :

```
Content_type: text/html
```

iar pentru un document fără comenzi HTML el trebuie să fie :

```
Content_type: text/plain
```

2) `Location: cale_spre_alt_URL \n\n`

aceasta corespunde cazului când scriptul nu a produs un document, ci o referință spre un URL (poate fi un document de pe același calculator, sau de pe altul). De exemplu un CGI de căutare în baze de date poate întoarce adresa URL-ului care conține informația căutată.

3) `Status:nnn Mesaj \n\n`

În acest caz, scriptul nu întoarce nimic altceva decît un mesaj (care poate fi de eroare sau de confirmare) și un cod numeric din 3 cifre.

4. ELEMENTE DE HTML

Serverul de web afișează o pagină interpretînd limbajul HTML (Hyper Text Markup Language) a cărei descriere completă este dincolo de scopul acestei lucrări, și din care vom prezenta cîteva elemente de bază.

Un document HTML conține două tipuri de elemente : instrucțiuni de formatare, numite tag-uri, și text. Tagurile nu sînt afișate de către Browser, ci îi indică acestuia cum să formateze textul.

Tag-urile sînt întotdeauna incluse între `<>`. De obicei efectul unui tag (să zicem `<tag>`) este anulat de `</tag>`. De exemplu pt a afișa un text în format italic se va folosi tagul `<I>` la începutul textului și `</I>` la sfîrșitul textului. Nu toate tagurile însă necesită și varianta `</tag>`. Cîteva din cele mai importante taguri:

- `<HTML>`, `</HTML>` încadrează documentul

- `<HEAD>`, `</HEAD>`, `<BODY>`, `</BODY>` incadrează antetul, respectiv corpul documentului; numai informația din corp va fi afișată
- `<TITLE>`, `</TITLE>` incadrează titlul documentului
- ``, `<I>`, `` (și cele de anulare corespunzătoare) incadrează textul ce se dorește a fi bold, italic sau underline
- `<P>`, `</P>` incadrează un paragraf; după `</P>` sau la întâlnirea unui nou `<P>` se trece la aliniat nou
- `
` face trecerea pe un rând nou
- `<HR>` introduce o linie orizontală
- ``, `` permit formatarea de liste; elementele listei se prefixează cu ``
- `` descriere `` introduce imaginea specificată, urmînd ca un browser ne-grafic sau unul cu imaginile dezactivate să scrie în schimb descrierea ca text
- `` Textul afișat la locul legăturii `` introduce o legătură (hyperlink)

Se poate vedea sursa oricărui document HTML folosind comanda browserului "View document source". Această comandă *nu* va arăta (evident) sursa scriptului CGI care a produs codul HTML, dacă acest cod este produs dinamic.

APLICAȚIE A5

Rulați din pagina de test "Afișează environment în Python". Folosind *view page source* urmăriți sursa HTML și identificați scriptul Python apelat. Urmăriți în sursa acestuia (aflată în copie în /asi/cgi-bin) cum folosește elemente de HTML.

Important pentru CGI: parametrii se pasează prin browser unui script folosind separatoarele ? și &, după URL (adresa scrisă în bara de adrese), de exemplu

<http://www.server.com/cgi-bin/script?param1=xxx¶m2=yyy>

APLICAȚIE A6

Rulați exemplul "Prelucrare QUERY_STRING în Python". Adăugați în URL-ul paginii care se deschide, diferiți parametri cum ar fi *param1=xxx¶m2=yyy* (șirul de parametri se adaugă după semnul întrebării). Urmăriți sursa Python. Cum împarte scriptul Python linia de parametri ? ce sintaxă HTML folosește pentru afișarea tabelului?

5. Scripturile *bash* folosite ca scripturi CGI

În continuare, se va examina "Un alt cgi" din pagina de test. Se vede cum, introducînd parametrul 1, 2 sau altă valoare în bara de adrese a browserului, de exemplu:

`http://matrix.elcom.pub.ro/cgi-bin/cgi?1`

se obțin comportamente diferite. Ceea ce se întîmplă este că scriptul rulează alte comenzi (programe) de pe server și afișează *output*-ul lor.

APLICAȚIE A7

Pasați parametrii 1, 2 sau 3 (sau alte valori) acestui script folosind „?”. Ce se întîmplă ? Examinați sursa scriptului.

6. FORMULARE HTML

Am spus că scripturile CGI pot fi folosite pentru prelucrarea unor date introduse de utilizator. Aceste date pot fi introduse printr-un element HTML numit formular.

Formularele pot prelua date de la user în mai multe moduri:

- Radio Buttons (butoane radio);

Acestea sînt casute care se pot “bifa” cu restricția că doar o căsuță poate fi bifată la un moment dat. Ele se definesc cu tagul <INPUT> sub forma :

```
<INPUT type="RADIO" name="NAME" value="VALOARE" CHECKED>
```

unde în câmpul “name” se introduce numele care se dorește pentru variabila respectivă, iar în câmpul “VALOARE” se introduce valoarea pe care o ia variabila dacă respectivă casuță este selectată. Cum o singură casuță poate fi selectată, cea care se dorește să fie activă implicit se marchează cu CHECKED.

- Check Boxes (căsuțe de selectare);

Sînt asemanătoare cu butoanele radio, doar că pot fi selectate mai multe la un moment dat, și au altă formă. Tagul este similar, doar câmpul “type” fiind diferit :

```
<INPUT type="CHECKBOX" name="NAME" value="VALOARE">
```

- Linie de text;

Aceasta permite utilizatorului să introducă un șir de caractere; sintaxa este :

```
<INPUT type="TEXT" name="NAME" value="VAL" size=nn maxlength=nn>
```

câmpul “value” este opțional; câmpurile maxlength și size permit specificarea lungii maxime, respectiv vizibile în cutie, a textului.

- TextArea (Cutie de text);

Similar cu tipul text, acest tip de control permite introducerea unui text format din mai multe linii:

```
<TEXTAREA name="NAME" ROWS=nn COLS=nn> bla bla bla </TEXTAREA>
```

dacă textul introdus depășește aceste dimensiuni (ROWS și COLS) vor apărea automat bare de derulare;

- Alte elemente:

- câmpul parola: similar cu text, doar că nu se afișează literele, ci stelute (*);

- lista: similar cu radiobuttons, doar că elementele se prezintă sub forma unei casute cu fiecare opțiune pe câte o linie; (consultați specificația HTML pentru detalii)

- Butoane :

Sînt două tipuri de butoane, ambele fiind de tipul INPUT.

- butonul RESET este folosit la resetarea tuturor parametrilor din formular la valorile implicite (dacă există); sintaxa este :

```
<INPUT type="RESET" value="RESET">
```

câmpul “value” de data aceasta este folosit pentru a specifica ce anume se afișează în buton. Acest buton nu trimite nici o informație serverului, acțiunea lui fiind în întregime pentru browser.

- butonul SUBMIT este folosit la încheierea redactării formularului, fiecare variabilă fiind transmisă către server împreună cu valoarea care i-a fost asignată de către user; sintaxa este :

```
<INPUT type="SUBMIT" value="OK" >
```

Acestea sînt elementele care pot compune un formular. Evident alături de ele se pot introduce și alte taguri HTML de exemplu pentru a scrie explicații în formular, cu diverse fonturi etc.

Formularul trebuie inclus între tagurile <FORM> și </FORM>; tagul <FORM> accepta parametrii :

```
<FORM ACTION="acțiune" METHOD="metoda">.
```

Acțiunea poate fi un nume de program sau script (CGI) care va primi controlul în momentul apăsării butonului SUBMIT. De asemenea ACTION poate fi un URL complet, formularul fiind trimis spre prelucrare în altă parte.

Metoda poate fi **GET** sau **POST**. Aceste două metode specifică modul în care se trimite informația către program, și anume:

- **POST** o trimite către intrarea standard a programului (stdin)
- **GET** o introduce în variabila de mediu QUERY_STRING (cu dezavantajul că dacă cantitatea de date este mare, o parte din date se pot pierde datorită trunchierii variabilei de mediu).

APLICAȚIE A8

Se examinează formularul “exemplu de formular” din pagina de test, precum și sursa acestuia (*view page source*)

7. APELAREA UNUI SCRIPT CGI

Un script CGI este introdus într-un document HTML similar cu o referință la un alt document, imagine, etc. Se poate folosi o sintaxă de genul :

```
<A HREF="/cgi-bin/myscript"> Ruleaza acest script </A>
```

dar și de genul

```

```

dacă scriptul va genera o imagine GIF, JPG etc. în urma apelării. Se observă deci că un script poate înlocui ORICE URL care nu este script, cu condiția ca să întoarcă informația în formatul în care se așteaptă de la respectivul URL. Pentru ultimul exemplu, să ne inchipuim că o anumită imagine nu este “statică” ci trebuie generată în funcție de anumite condiții (de exemplu, un grafic). Atunci nu se cheamă direct imaginea, ci un script care generează imaginea după necesități.

APLICAȚIE A9

Identificați scriptul cgi apelat de “exemplu de formular” și citați sursa Python a acestuia.

Încercați rularea acestui script direct din terminal:

```
./cgi-list.py
```

Ce se observă?

Observați că pentru citirea datelor introduse de user în formular se folosesc:

```
form=cgi.FieldStorage()  
colors=form.getlist('color')
```

Examinați și aplicația „conversie temperatură” care folosește tot un câmp de formular.

APLICAȚIE A10 *

Să se scrie un script CGI în Python care să permită conversia temperaturii din F în C, similar cu cel existent (al cărui sursă nu este disponibilă).

Indicații

- Pentru a avea permisiunile necesare, aveți deja drept de scriere pentru câte un fișier de pornire `cgiN.py` în fiecare director `/asi/u1...uN` corespunzător numărului echipei N, cu link în `cgi-bin`-ul serverului pt a fi executabil; în varianta „goală”, acest script întoarce textul „nimic”. *Modificați acest fișier, dar NU-L STERGETI* căci nu îi mai puteți pune la loc permisiunile (veți fi penalizați dacă îl stergeți).
- Observați numele câmpului de input în care userul introduce temperatura, examinând sursa HTML a formularului (*view page source*)
- Examinați sursa lui `cgi-list.py` pentru a vedea cum se inițializează *form* folosind `cgi.FieldStorage()`. Sursa voastră va fi similară, doar că veți citi un singur input de tip text, nu mai multe butoane.
- În Python citiți valoarea un *singur* câmp XXX folosind `form.getfirst('XXX')`; convertiți în tipul de date “int” și aplicați formula aritmetică de conversie.
- Nu uitați să generați și headerul HTML cu o linie goală după Content-Type ! Vedeți detalii despre protocolul HTTP și în cursul despre HTTP de pe site.

Bibliografie

- [1] CGI support in Python: <https://docs.python.org/3/library/cgi.html>
- [2] CGI Tutorial: <http://cgi.tutorial.codepoint.net>
- [3] Web Programming with Python - John Loomis www.johnloomis.org/python/