Lucrarea 4
HTTP şi CGI

1. NOTIUNI INTRODUCTIVE

CGI (Common Gateway Interface) reprezintă o interfaţă (un set de norme) care fac posibilă comunicarea între un server de HTTP şi un program care ruleaza pe acelaşi calculator cu serverul. Prin CGI se poate apela programul de către server, i se pot transmite parametri, şi se poate procesa output-ul produs de program.

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

- citirea şi 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 citeasca "intrarea standard" (stdin) sau variabilele de mediu (environment) şi să scrie la "iesirea standard" (stdout).

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

NOTA2 : Exista şi alte moduri de a genera continut dinamic într-o pagina de web. CGI se foloseste mai ales atunci cînd se doreste interacţiunea cu un program extern.

Reamintim semnificaţia conceptelor de *stdin, stdout* şi *environment*.

- "*stdin*" şi "*stdout*", prescurtari de la "*standard input*" şi "*standard output*" reprezintă "sursa" din care programul citeste datele de intrare, respectiv "destinatia" la care scrie datele de iesire. În mod obisnuit, pentru un program lucrînd în mod text (la consola DOS/Windows sau UNIX), stdin este tastatura, iar stdout este fereastra unde se afişează. Acestea sînt "redirectionate" automat prin TCP/IP atunci cînd nu se lucrează direct de la consolă, ci de exemplu prin conectarea cu ssh. Dacă se doreste înlocuirea stdout cu altceva, de exemplu o comanda „ls" se doreste să scrie într-un fişier şi nu pe ecran, se foloseste operatorul de indirectare notat ">", cu sintaxa:

```
ls                          afişează rezultatul pe ecran
ls > fisier                 scrie rezultatul în fisier
ls /bin > fisier            scrie rezultatul în fisier
ls /bin | cat > fisier      acelaşi lucru „mai complicat"; semnul | se citeşte „pipe" (conductă);
```

stdout-ul comenzii ls este automat vărsat în stdin-ul comenzii cat, iar acesta concatenează (copiază) în fisier

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

lui tail)

La fel pentru citirea din altă parte decît tastatura, se foloseste indirectarea dintr-un fişier folosind "<".

- "*environment*" (mediu/variabile de mediu) se referă la o modalitate a sistemului de operare de a comunica între programe; o variabilă de environment se poate seta de către un program sau de către utilizator şi poate fi citită de alte programe. De exemplu în DOS şi UNIX exista o variabilă de environment numită PATH, în care se stochează directoarele (folderele) în care să fie căutate implicit programele executabile. PATH se setează 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 functii de sistem (în C functia este "*getenv()*") care să intoarca valoarea lor, spre a le folosi (astfel sistemul de operare stie unde să caute executabilele mai importante, programele stiu unde să-şi puna fişierele temporare, etc).

Alte comenzi care permit lucrul cu variabile de mediu:

DOS/Windows	UNIX	descriere
`env` (doar în 2000/XP)	`env`	vizualizeaza toate variabilele de mediu (*env*ironment)
`path`	`echo $PATH` sau `env | grep PATH`	vizualizeaza continutul variabilei PATH

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

APLICAŢIE A1

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

- creaţi folosind *export* o variabilă de mediu cu o valoare la întîmplare, şi afişaţi-i valoarea. De obicei variabilele se denumesc cu litere mari. Atenţie, nu lăsaţi spaţiu în stînga şi dreapta semnului egal!

```
export VAR1=111
env
env | grep VAR1
```

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

Consecinţă: serverul web este un proces, şi procesele pe care le crează el vor moşteni environmentul creat de acesta !

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

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

APLICAŢIE A2

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

- Ce comandă veţi da pentru a afişa doar procesele care aparţin userului curent ? observaţi că nu este suficient să puneţi doar numele userului în grep, căci acest nume poate apărea şi pentru procese care nu-i aparţin acestuia.

2. VARIABILE DE ENVIRONMENT SPECIFICE CGI

În momentul în care serverul lansează în executie un script CGI, el setează un număr de variabile de environment pe care scriptul le poate examina. Uneori scriptul preia toata informaţia de care are nevoie din variabilele de mediu, alteori din intrarea standard sau linia de comanda, sau alteori 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 copiati-l în home-ul vostru). Evident, o modificare a scripturilor din /asi/cgi-bin nu va duce și la modificarea originalelor.

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

APLICAŢIE 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; rulați scriptul:

```
./test-cgi
```

Urmăriți rezultatul afișat. De ce variabilele au valori diferite în browser (care afişează pagina de pe matrix) şi în terminalul (*tot pe matrix*), din moment ce este vorba de acelaşi 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, pentru CGI ne interesează în special:

- QUERY_STRING

  Conţine efectiv parametrii cu care a fost "chemat" scriptul, sub forma unui şir cu parametrii separati prin semnul "&"; Acesta este sirul care a fost "pasat" scriptului dupa semnul "?". Tipic, şirul 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".

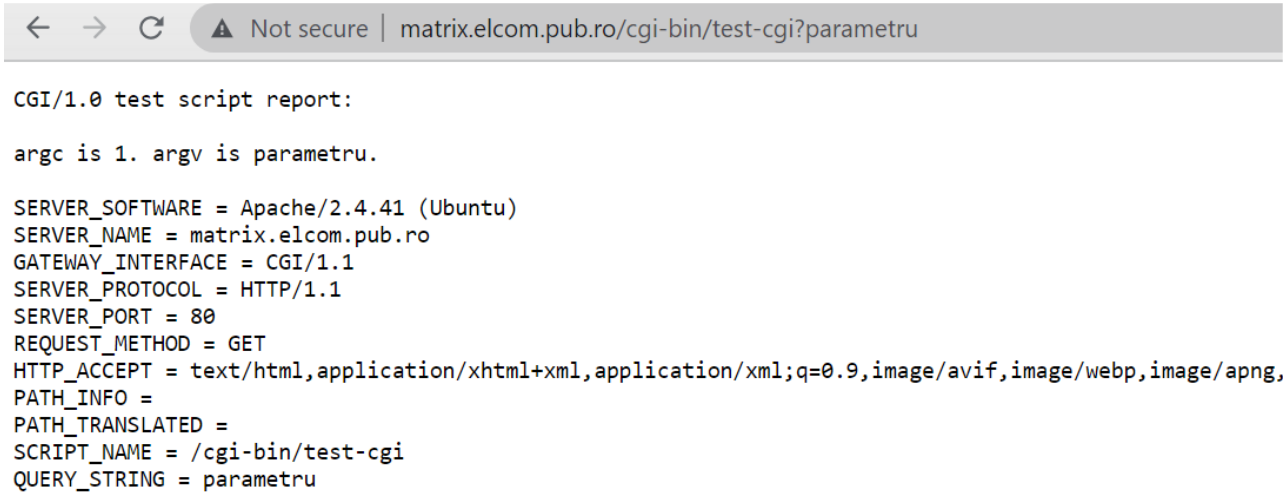
- CONTENT_TYPE
  Pentru cererile care au ataşată o informaţie, cum ar fi HTTP POST, aceasta variabila specifica tipul datelor.

- CONTENT_LENGTH

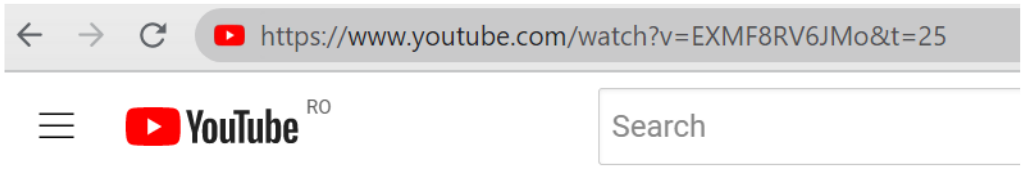
Conţine lungimea datelor trimise de client; astfel, browserul va sti cîte caractere să citeasca din *query-string*.

APLICAŢIE A4

Cum puteti modifica valoarea variabilei QUERY_STRING din environmentul afişat în browser în pagina de test? dar cea din terminal? Ce *metodă* este folosită în cîmpul REQUEST_METHOD? țineți minte această metodă, va fi folosită și ulterior, la pasarea parametrilor prin URL. Indicație: în bara de adrese din browser, semnul ? separă URL-ul de parametrii care urmează.



Parametrii respectivi se pot adăuga manual în linia de comandă, separați cu &. Aceeași situație este folosită, de exemplu, cînd adăugați momentul de timp în care doriți să înceapă redarea unui videoclip youtube, sub forma parametrului t=25 (în secunde):



3. PRODUCEREA DE INFORMAŢII (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 greşeală frecventă este generarea directă de către script a unui document HTML, ca şi cînd acest document ar fi scris de mînă și plasat într-un folder al serverului. De fapt, pentru ca serverul să înţeleagă despre ce tip de document este vorba, trebuie generat mai întîi 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" şi "subtip" sînt tipuri standard, Iar \n\n semnifica lasarea unei linii goale dupa 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 cînd scriptul nu a produs un document, ci o referinta spre un URL (poate fi un document de pe acelasi calculator, sau de pe altul). De exemplu un CGI de cautare în baze de date poate intoarce adresa URL-ului care conţine informaţia cautata.

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

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

4. ELEMENTE DE HTML

Serverul de web afiseaza o pagina interpretînd limbajul HTML (Hyper Text Markup Language) a carui descriere completă este dincolo de scopul acestei lucrari, şi din care vom prezenta cîteva elemente de bază.

Un document HTML conţine doua tipuri de elemente : instructiuni de formatare, numite tag-uri, şi text. Tagurile nu sînt afisate de către Browser, ci îi indica acestuia cum să formateze textul.

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

- <HTML>, </HTML> incadreaza documentul
- <HEAD>, </HEAD>, <BODY>, </BODY> incadreaza antetul, respectiv corpul documentului; numai informaţia din corp va fi afisata
- <TITLE>, </TITLE> incadreaza titlul documentului
- <STRONG>, <I>, <UL> (şi cele de anulare corespunzatoare) incadreaza textul ce se doreste a fi bold, italic sau underline
- <P>, </P> incadreaza un paragraf; dupa </P> sau la intilnirea unui nou <P> se trece la aliniat nou
-
 face trecerea pe un rînd nou
- <HR> introduce o linie orizontala
- <OL>, <UL> permit formatarea de liste; elementele listei se prefixează cu <LI>
- <IMG SRC=”fişier.gif”> descriere </IMG> introduce imaginea specificata, urmind ca un browser ne-grafic sau unul cu imaginile dezactivate să scrie în schimb descrierea ca text
- <A HREF=”legatura”> Textul afisat la locul legaturii </A> introduce o legatura (hyperlink)

Se poate vedea sursa oricărui document HTML folosind comanda browserului “View document source”. Aceasta comanda *nu* va arata (evident) sursa scriptului CGI care a produs codul HTML, dacă acest cod este produs dinamic.

APLICAŢIE 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 elmente de HTML.

Observați cum parametrii se pasează prin browser unui script folosind separatoarele ? și &

http://www.server.com/cgi-bin/script?*param1=xxx&param2=yyy*

APLICAŢIE A6

Rulați exemplul "Prelucrare QUERY_STRING în Python". Adăugați în URL-ul paginii care se deschide, diferiți parametri cum ar fi *param1=xxx&param2=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 obţin comportamente diferite. Ceea ce se întîmplă este că scriptul rulează alte comenzi (programe) de pe server şi afişează *output*-ul lor.

APLICAŢIE 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 restrictia 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 cîmpul "name" se introduce numele care se doreste pentru variabila respectiva, iar în cîmpul "VALOARE" se introduce valoarea pe care o ia variabila dacă respectiva casuţă este selectata. Cum o singuţă casuta poate fi selectata, cea care se doreste să fie activa implicit se marcheaza cu CHECKED.

- Check Boxes (căsuţe de selectare);
  Sînt asemanatoare cu butoanele radio, doar că pot fi selectate mai multe la un moment dat, şi au altă forma. Tagul este similar, doar cîmpul "type" fiind diferit :

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

- Linie de text;
  Aceasta permite utilizatorului să introduca un sir de caractere; sintaxa este :

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

  cîmpul "value" este optional; cîmpurile maxlength şi 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 şi COLS) vor apare automat bare de derulare;

- Alte elemente:
  - cîmpul parola: similar cu text, doar că nu se afiseaza literele, ci stelute (*);
  - lista: similar cu radiobuttons, doar că elementele se prezinta sub forma unei casute cu fiecare optiune pe cîte o linie; (consultati specificatia HTML pentru detalii)
- Butoane :

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

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

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

cîmpul "value" de data aceasta este folosit pentru a specifica ce anume se afiseaza în buton. Acest buton nu trimite nici o informaţie serverului, acţiunea lui fiind în intregime pentru browser.

- butonul SUBMIT este folosit la incheierea redactarii formularului, fiecare variabila fiind transmisa către server impreuna cu valoarea care i-a fost asignata 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 şi alte taguri HTML de exemplu pentru a scrie explicații în formular, cu diverse fonturi etc.

Formularul trebuie inclus între tagurile <FORM> şi </FORM>; tagul <FORM> acceptă parametrii :

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

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

Metoda poate fi **GET** sau **POST**. Aceste două metode specifică modul în care se trimite informaţia către program, şi 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 datorita trunchierii variabilei de mediu).

APLICAŢIE A8

Se examinează formularul "exemplu de formular" din pagina de test, precum si sursa acestuia (*view page source*). Ce se întîmplă la apăsarea butonului Submit? Ce metodă (GET sau POST) se folosește?

7. APELAREA UNUI SCRIPT CGI

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

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

dar şi de genul

```
<img src="http://www.blabla.com/cgi-bin/myscript">
```

dacă scriptul va genera o imagine GIF, JPG etc. în urma apelarii. Se observă deci că un script poate inlocui ORICE URL, cu condiția ca să întoarcă informația în formatul în care se așteaptă de la respectivul URL. Pentru ultimul exemplu, avem cazul cînd o anumita imagine nu este statică, ci trebuie generata în funcție de anumite condiții (de exemplu, un grafic de temperatură). Atunci nu se cheamă direct imaginea, ci un script care generează imaginea pe baza datelor selectate de user.

APLICAȚIE A9

Identificați scriptul cgi apelat de "exemplu de formular" folosind *View Page Source* si citiți 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 cîmp de formular.

Pînă acum toate exemplele HTML au fost în directorul ***/var/www/html/****asi* (puteți să verificați din contul vostru). Voi ați introdus în browser doar adresa http://matrix.elcom.pub.ro/asi/.... deci primele 3 directoare în arbore, pînă la „asi", au fost luate automat de server, pentru că așa e configurat „document root". În acest director, userii care nu sînt root nu pot scrie.
Un user poate publica pe web PROPRIUL fișier HTML cu condiția să-l pună într-un folder numit *public_html* aflat în home-directory-ul său (atenție la sintaxă! browserul afișează pagini doar dintr-un folder EXACT cu acest nume). De asemenea poate rula scripturi CGI din același folder (evident toate aceste căi și permisiuni se setează în configurația serverului).
Mai departe, pt. a vedea pe web fișierul *fisier.html* unui user numit asiNN, trebuie introdusă în brower adresa:
http://matrix.elcom.pub.ro/~asiNN/fisier.html
(observați semnul ~ care aici se citește „home").

APLICAȚIE A10 *

Să se scrie un script CGI în Python care să permită conversia temperaturii din F în C, preluată dintr-un formular aflat pe o pagină HTML din contul personal, care, de asemenea, trebuie creată. El trebuie să se comporte similar cu cel existent „conversie temperatură" (a cărui sursă nu este disponibilă în /asi/cgi-bin).

Indicații

- Veți scrie scriptul CGI în directorul corespunzător echipei într-un fișier numit */home/asi/asiNN/public_html/apli10.cgi* iar fișierul HTML va fi */home/asi/asiNN/public_html/apli10.html* ; pentru a-l vedea în browser folosiți prescurtarea de mai sus, cu „~". Fișierul HTML îl faceți similar cu sursa lui „Conversie temperatură" de pe pagina de test (view page source), doar că îl loc de *action="/cgi-bin/temperatura.py"* veți pune *action="http://matrix.elcom.pub.ro/~asiNN/apli10.py"* căci scriptul vostru se află acolo și nu în folderul /cgi-bin de sistem.

- Observați numele cîmpului de input în care userul introduce temperatura, examinînd 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* cîmp XXX folosind form.getfirst('XXX'); convertiți în tipul de date "int" și aplicați formula aritmetică de conversie a temperaturii.

- Nu uitați sa generati si headerul HTML cu o linie goală după Content-Type așa cum fac și celelalte scripturi date ca exemplu! 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/*