

# Laboratorul 12 – Protocolul web HTTP.

**Responsabil:** Ciprian Dobre <cipsm@cs.pub.ro>

**Data publicării:** 18-05-2008

**Data ultimei modificări:** 18-05-2008

## ***Obiective***

În urma parcurgerii acestui laborator studentul va fi capabil să:

- Întelegea functionalitatea protocolului HTTP.
- Să întelegea rolul entitatilor implicate într-un schimb de mesaje HTTP.
- Să scrie un client simplu de HTTP.

## ***Prezentarea laboratorului***

HTTP (HyperText Transfer Protocol - protocol pentru transfer de hipertext) este protocolul standard folosit în Web. HTTP este un protocol ASCII simplu. O interacțiune HTTP constă dintr-o cerere de tip text, urmată de un răspuns care constă în fișierul transferat (ca mesaj în format standard MIME). Se poate spune că protocolul HTTP este format din multimea cererilor (uzual GET, POST) de la programele de navigare către servere și din multimea răspunsurilor trimise de acestea (antetele și corpurile mesajelor sunt asemănătoare cu cele din standardul MIME).

HTTP cunoaște mai multe versiuni succesive și este în continuă evoluție; ultimele versiuni de MIME sunt foarte flexibile, având o deschidere pentru viitoare aplicații obiectuale.

HTTP este utilizat de către navigatoare, dar ar putea fi folosit și de către o persoană aflată la un terminal pentru a "discuta" direct cu un server Web, folosind o conectare TCP (de exemplu, prin programul telnet).

Formatul unui mesaj HTTP este:

```
<linia initiala, diferita pentru cerere si raspuns >  
Header1: value1  
Header2: value2  
Header3: value3  
  
<corp>
```

Linia initiala este diferita pentru cerere și răspuns. O cerere tipică de tip GET arată astfel:

```
GET /path/to/file/index.html HTTP/1.0
```

iar linia initiala a unui raspuns arata:

```
HTTP/1.0 200 OK
```

sau

```
HTTP/1.0 404 Not Found
```

## ***Comunicarea cu serverul***

### **Intrarea**

Modul prin care primim informatiile de la server este prin "variabile de mediu": o serie de perechi de forma (nume, valoare). Structura (nume, valoare) a variabilelor de mediu se potriveste foarte bine cu datele care vin din formularul (form) pe care utilizatorul l-a incarcat in browser. Fiecare camp de intrare de pe formular poarta un nume si va primi si o valoare furnizata de catre utilizator. Valoarea trebuie sa fie legata de numele campului si perechea sa fie trimisa serverului Web.

Exista doua variante de a trimite catre serverul Web informatia culeasa intr-un formular care apeleaza un script CGI: GET si POST. Formularul (form) este o pagina web in care se afla un tag <form> </form> in care se pot introduce date prin intermediul marcatorilor <input>, <select> s.a. Pentru detalii in ceea ce priveste formurile HTML cititi <http://www.w3.org/TR/REC-html40/interact/forms.html>.

Fiecare informatie dintr-un formular este codificata astfel: se fac perechi de (informatie, valoare), separate între ele prin &, spatiile se inlocuiesc cu + si orice alt caracter care nu este alfanumeric se inlocuieste cu % si codul hexa de doua cifre.

In cazul metodei GET informatiile codificate vor aparea în URL-ul scriptului care prelucreaza informatia, totodata va fi pus în variabila de mediu QUERY\_STRING. Dar aceasta are dezavantajul unei posibilitati de trimitere limitata de informatie (maxim 1k) si in plus utilizatorul va vedea datele introduse, ceea ce nu e bine, de exemplu sa vada un URL cu parola sau cu seria cartii de credit. La POST in schimb datele se pun la intrarea standard de unde le poate lua programul, sub forma unui fisier codat cu standardul MIME inglobat in corpul mesajului trimis catre server. Datorita faptului ca datele sunt trimise codat MIME, serverul stie marimea si tipul datelor si va trece aceste valori in script folosind variabila de mediu CONTENT\_LENGTH si CONTENT\_TYPE.

De exemplu, un formular cu metoda POST care contine doua variabile NAME si PHONE va trimite data astfel: NAME=Ana&PHONE=4556. Metoda GET va adauga acest sir de caractere la URL, dupa semnul de intrebare.

Pentru a vedea daca formul a folosit metoda GET sau POST, programul CGI trebuie sa se uite in variabila de mediu REQUEST\_METHOD.

**Observatie:** La primirea datelor, acestea sunt codificate asa cum s-a specificat mai sus, iar programul CGI trebuie sa faca operatia inversa pentru decodificare. Pentru a citi/seta variabilele de mediu se folosesc functiile :

```
char *getenv(char *env_var) Returneaza valoarea variabilei de mediu  
specificata de env_var  
char *setenv(char *env_var, char *env_value) Seteaza valoarea  
variabilei de mediu specificata de env_var cu valoarea specificata de  
env_value.
```

## Iesirea

In momentul in care serverul HTTP primeste o cerere care se refera la un program CGI, serverul va executa programul oferindu-i datele de intrare necesare, iar output-ul programului va fi trimis la client. Output-ul programului reprezinta tot ce acesta afiseaza in mod normal pe stdout. In acest caz va afisa spre client. Totusi, datele trimise de catre programul CGI trebuie sa fie codificate prin standardul MIME, astfel incat un programul CGI trebuie sa porneasca prin trimiterea unui antet (header) MIME care specifica tipul datelor care vor urma, antet care este separat de informatia codata printr-un rand liber.

## Exemplu de headere

```
Set-Cookie: UserID = MadJad; Expires = Tuesday, 09-Aug-05 11:34:00 GMT;  
Location: ../images/picture1.jpg  
Content-Type: text/html
```

Trebuie sa includeti unul (sau mai multe) din aceste headere in programul vostru pentru a comunica serverului HTTP ce trebuie sa faca dupa ce ruleaza programul CGI.

Acestea se trimit pur si simplu cu

```
printf("Content-type: text/html\n\n").
```

Prin linia de mai sus s-a specificat faptul ca datele care urmeaza reprezinta cod HTML.

Dupa acest printf se pot trimite date de tip HTML.

**Observatie** : in momentul in care trimiteti headere nu neglijati cei doi "\n" care sunt importanti pentru comunicarea corecta.

## HTTP State Management - cookie-uri

Mecanismul de cookie reprezinta principalul mecanism HTTP de mentinere a starii de-a lungul mai multor cereri (request) si raspunsuri (response). In mod obisnuit, serverele HTTP raspund la cererile clientilor in mod individual, fara a pastra vreo legatura cu cererile anterioare sau ulterioare. Pentru a pastra astfel de legaturi, s-au introdus doua headere speciale HTTP: *Set-Cookie* si *Cookie*.

Mecanismul de management al sesiunii HTTP functioneaza in felul urmator:

1. Serverul initiaza o sesiune trimitand in raspunsul spre client un header *Set-Cookie*.
2. Clientul poate accepta sau respinge cookie-ul trimis de server pe baza atributelor acestuia. Un cookie este respins daca a expirat sau daca domeniul (*Domain*) sau calea (*Path*) nu corespund URL-ului curent. In cazul in care cookie-ul este acceptat si clientul doreste continuarea sesiunii, acesta va trimite un header *Cookie* in cererile ulterioare.
3. Serverul poate ignora header-ul *Cookie* primit de la client sau il poate folosi pentru a determina starea curenta a sesiunii. Comunicatia poate continua, serverul trimitand un nou header *Set-Cookie* s.a.m.d.
4. Sesiunea este terminata prin trimiterea unui *Set-Cookie* cu *MaxAge=0* (vezi mai jos).

Headerul *Set-Cookie*: are urmatorul format:

*Set-Cookie: Nume=Valoare; attribute [, Nume=Valoare; attribute ... ]*

Fiecare cookie are asociat un nume si o valoare (de tip string). Principalele attribute sunt:

- *Domain="domain"* - cookie-ul este valabil doar pentru host-urile din domeniul specificat. "domain" trebuie sa inceapa cu punct.
- *Path="path"* - cookie-ul este valabil doar pentru URL-urile care incep cu /path
- *Max-Age="seconds"* - timpul de valabilitate al cookie-ului, in secunde. Dupa ce acest timp expira, clientul trebuie sa stearga/ignore cookie-ul.
- *Version="ver"* - versiunea pentru cookie-uri. In implementarea curenta, versiunea folosita este 1.

Headerul *Cookie*: are un format similar:

*Cookie: \$Version="ver"; Nume=Valoare [; \$Path="path" ] [; \$Domain="domain"]*

,unde Version, Path si Domain au aceeasi semnificatie ca mai sus.

*Exemplu (cos de cumparaturi):*

Sens Comunicatie	Headere	Semnificatie
Client -> Server	POST /blah/login HTTP/1.1 [form data]	Utilizatorul se autentifica
Server -> Client	HTTP/1.1 200 OK Set-Cookie: Customer="POPESCU_ION"; Version="1"; Path="/blah"	Cookie-ul reflecta identitatea utilizatorului

Client -> Server	POST /blah/pickitem HTTP/1.1 Cookie: \$Version="1"; Customer="POPESCU_ION"; \$Path="/blah" [form data]	Utilizatorul a selectat un produs si il adauga in cosul de cumparaturi.
Server - Client	HTTP/1.1 200 OK Set-Cookie: Part_Number="Rocket_Launcher_0001"; Version="1"; Path="/blah"	Cosul de cumparaturi contine un produs.

## ***Aplicație***

Scrieti un program client care se conecteaza la un server web si cere o pagina folosind protocolul HTTP. Aplicatia va citi adresa serverului si numele fisierului html cerut de la tastatura.

*Indicatii:*

Programul se va conecta pe TCP la adresa citita si va trimite o linie text continand:

GET /file\_name HTTP/1.1\n\n

Atentie: numele trebuie specificat in format absolute; cererea e terminate de trimiterea unei linii goale (vedeti cei doi \n)

Ulterior aplicatia va citi raspunsul (pagina html intoarsa).

## ***Resurse:***

- <http://www.w3.org/TR/REC-html40/interact/forms.html>.
- [RFC 2616 - HTTP/1.1](#)
- <http://hoohoo.ncsa.uiuc.edu/cgi/overview.html>
- [RFC 2109 - HTTP Cookies](#)