Programare Web

Curs:
Florin Radulescu
Ciprian Dobre

Administrative

Notare

- Nota finala la cursul de **Programare Web** se calculeaza dupa urmatoarea formula:

Nota = ROUND(NotaCurs*0,15 + NotaLaborator*0,45 + NotaExamen*0,4)

unde:

- NotaCurs, NotaLaborator sunt obtinute pentru activitatile din timpul semestrului;
- NotaExamen este nota obtinuta la examen.
- Examenul este promovat **numai daca** se obtine minimum 50% atat din nota de la examenul final cat si din punctajul din timpul semestrului.

Notare

- NotaCurs se calculeaza pe baza
  - activitatii la curs (prezenta),
  - a unor *teste* (neanuntate) si a unor
  - *teme de casa*.
- Temele de casa date la curs – de dificultate medie - se predau la cursul imediat urmator celui in care au fost propuse si nu se refac.

Notare

- NotaLaborator se calculeaza pe baza
  - notelor la temele de laborator,
  - a activitatii de la laborator si
  - a task-urilor propuse dupa fiecare laborator
- Se tine cont nu doar de prezenta fizica in laborator, ci este notata mai ales implicarea studentului in realizarea lucrarilor practice din fiecare saptamana.
- Regulamentul de curs si laborator se va afisa pe situl cursului (de la curs.cs.pub.ro)

Participarea la ore

- Fiecare student participa la curs conform seriei in care este inscris.
- Fiecare student participa la laborator conform grupei din care face parte.
- Participarea cu alta serie / grupa este posibila in cazuri bine justificate, cu acordul cadrelor didactice implicate.

Capitolele cursului

Capitole - FR

1. Arhitectura unei aplicatii web
2. Servere HTTP: caracteristici, exemple
3. Limbajul HTML: elemente de baza, tabele, cadre, formulare
4. Scripturi CGI. Scripturi scrise in limbaje compilate.
5. Folosirea limbajului SQL pentru programarea web. ODBC
6. Limbaje de scripting server-side, PHP

Capitole - CD

7. Programare web client-side, jscript
8. Continut si design in programarea web, CSS, DHTML
9. Elemente multimedia pentru programarea web
10. Administrarea continutului site-urilor Web: optimizarea paginilor web pentru indexarea de catre motoarele de cautare, solutii profesionale
11. Elemente de securitate si optimizare

Capitolul 1. Arhitectura unei aplicatii web

Evolutie

- Arhitectura unei aplicatii informatice folosita intr-o organizatie a trecut prin mai multe etape:

1. Etapa Mainframe
2. Etapa Client-Server
3. Etapa Web Based (aplicatii web sau web-enabled)

Etapa Mainframe

- Un calculator de tip Mainframe.
- O multitudine de terminale (dumb terminals) conectate la acesta.
- Aplicatiile erau centralizate (rulau pe sistemul mare).
- Nu se punea problema arhitecturii aplicatiei decat in termenii de mai sus.
- Nu existau de obicei conexiuni din afara organizatiei -> securitatea nu era o problema majora.

Etapa client-server

- Un calculator (uneori mai multe) de tip minicalculator pe care ruleaza serverul.
- O serie de PC-uri pe care ruleaza programele client (programe dedicate furnizate o data cu aplicatia).
- Retea locala prin care comunica masinile respective.
- Apare si posibilitatea calculului distribuit (distributed computing) favorizat de vitezele din ce in ce mai mari de comunicatie

Etapa client-server

- Softul de baza (SO, SGBD) se adapteaza si el pentru lucrul in retea si calcul distribuit.
- Problemele noi care apar sunt legate de:
  - Conectivitate
  - Performanta
  - Securitate

Etapa client-server

- In prima perioada a etapei client-server aplicatiile erau mai ales departamentale:
- In cadrul organizatiei fiecare departament (productie, financiar, resurse umane, relatii cu clientii, etc) avea propria sa aplicatie accesata in reteaua locala a departamentului.

Etapa client-server

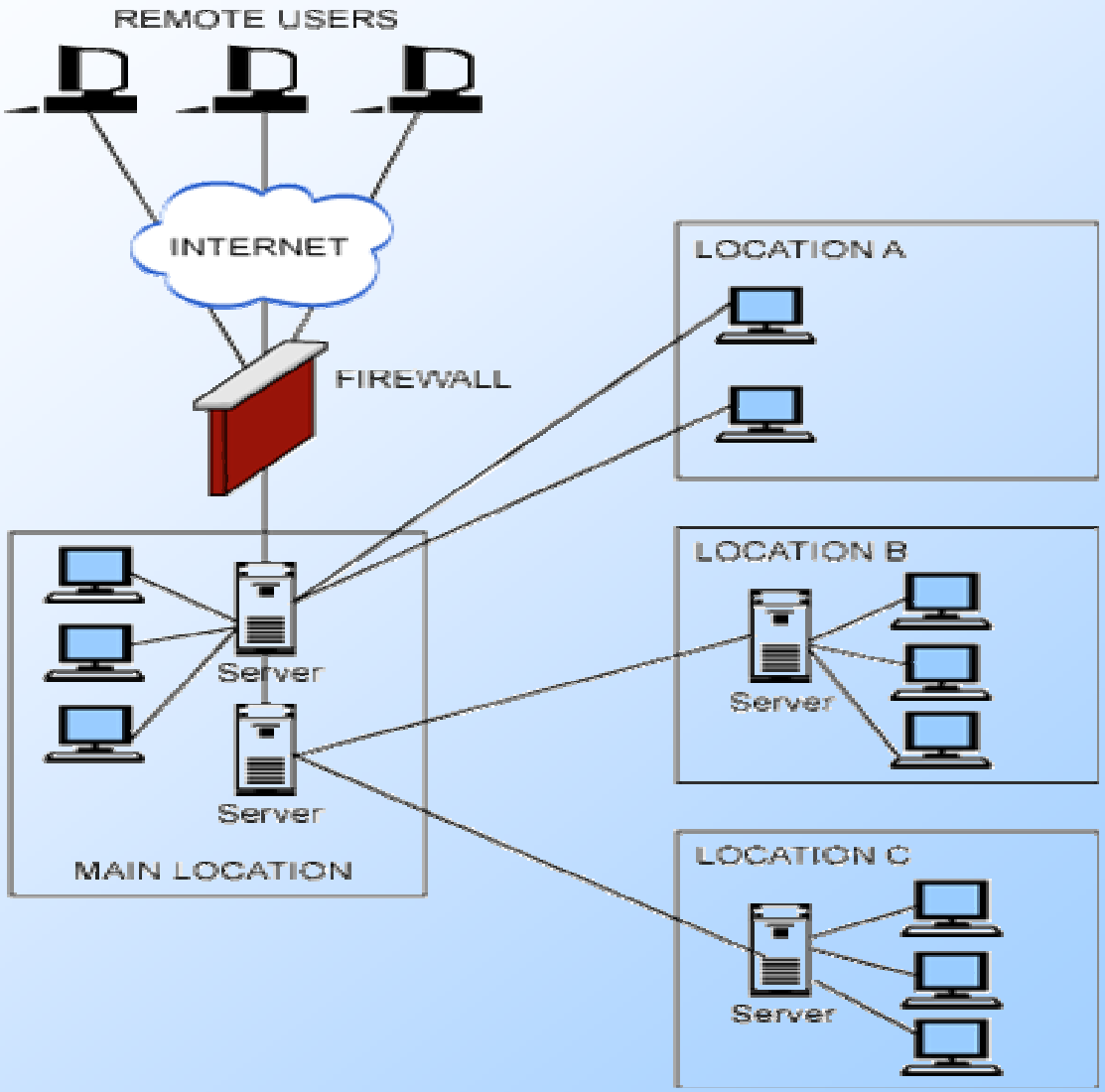
- Din aceasta cauza problemele de conectivitate si securitate nu erau critice, cele mai importante fiind problemele de performanta.
- Ca urmare, de cele mai multe ori responsabilitatile unui astfel de sistem erau cumulate de catre administratorul bazei de date (principalele elemente de performanta ale unei aplicatii erau date de performantele bazei de date)

Etapa client-server

- Aparitia sistemelor de tip ERP (Enterprise Resource Planning – sistem informatic integrat de intreprindere) a dus la necesitatea interconectarii si integrarii aplicatiilor departamentale intr-un tot unitar.
- In acel moment problemele de securitate si de conectivitate au capatat o mult mai mare importanta.

Etapa web-based

- Deosebirea principala in acest caz este accesarea aplicatiei
  - Printr-un client standard (browserul)
  - De foarte multe ori din afara locatiei unde se afla aplicatia
  - Comunicatia se face nu prin canale private / dedicate ci prin Internet
- O arhitectura tipica este in figura urmatoare [1]:


REMOTE USERS
INTERNET
FIREWALL
LOCATION A
LOCATION B
Server
Server
MAIN LOCATION
Server
LOCATION C
Server

Probleme de rezolvat

- In care locatii trebuie sa existe server (= bani necesari dotarii cu asa ceva) si in care nu. Pentru asta se analizeaza:
  - Numarul de utilizatori dintr-o anumita locatie
  - Latimea de banda disponibila intre locatia respectiva si locatia principala
  - Timpul de raspuns necesar de asigurat utilizatorilor

Probleme de rezolvat

- Rezolvarea problemelor de securizare a transmisiei de date: datele circuland in afara retelei organizatiei trebuie de exemplu criptate corespunzator.
- Problemele legate de browser: aplicatia trebuie sa poata fi accesata in acelasi mod si fara pierderi de functionalitate prin oricare din tipurile principale de browser existente.

Probleme de rezolvat

- Probleme legate de conectivitate: din toate locatiile de unde aplicatia e necesar sa fie accesibila timpul de raspuns trebuie sa fie in parametrii ceruti.
- De asemenea trebuie apreciat traficul generat de o aplicatie in contextul in care aceasta ruleaza in paralel cu alte aplicatii. Performantele pot fi afectate in cazul in care traficul cumulat depaseste anumite praguri.

Cum continuam?

- In partea a doua a cursului veti studia dezvoltarea de aplicatii web complexe.

- In prima parte insa ne vom concentra pe elementele de baza: HTML, CGI, folosirea bazelor de date in aplicatii web si limbajul de scripting PHP.

Cum continuam?

- In partea a doua a cursului veti studia dezvoltarea de aplicatii web complexe.

- In prima parte insa ne vom concentra pe elementele de baza: HTML, CGI, folosirea bazelor de date in aplicatii web si limbajul de scripting PHP.

Exemplu de arhitectura

- In cele ce urmeaza prezentam arhitectura unei aplicatii web simple in care datele sunt stocate intr-o baza de date relationala.
- Actorii implicati intr-o astfel de aplicatie sunt:

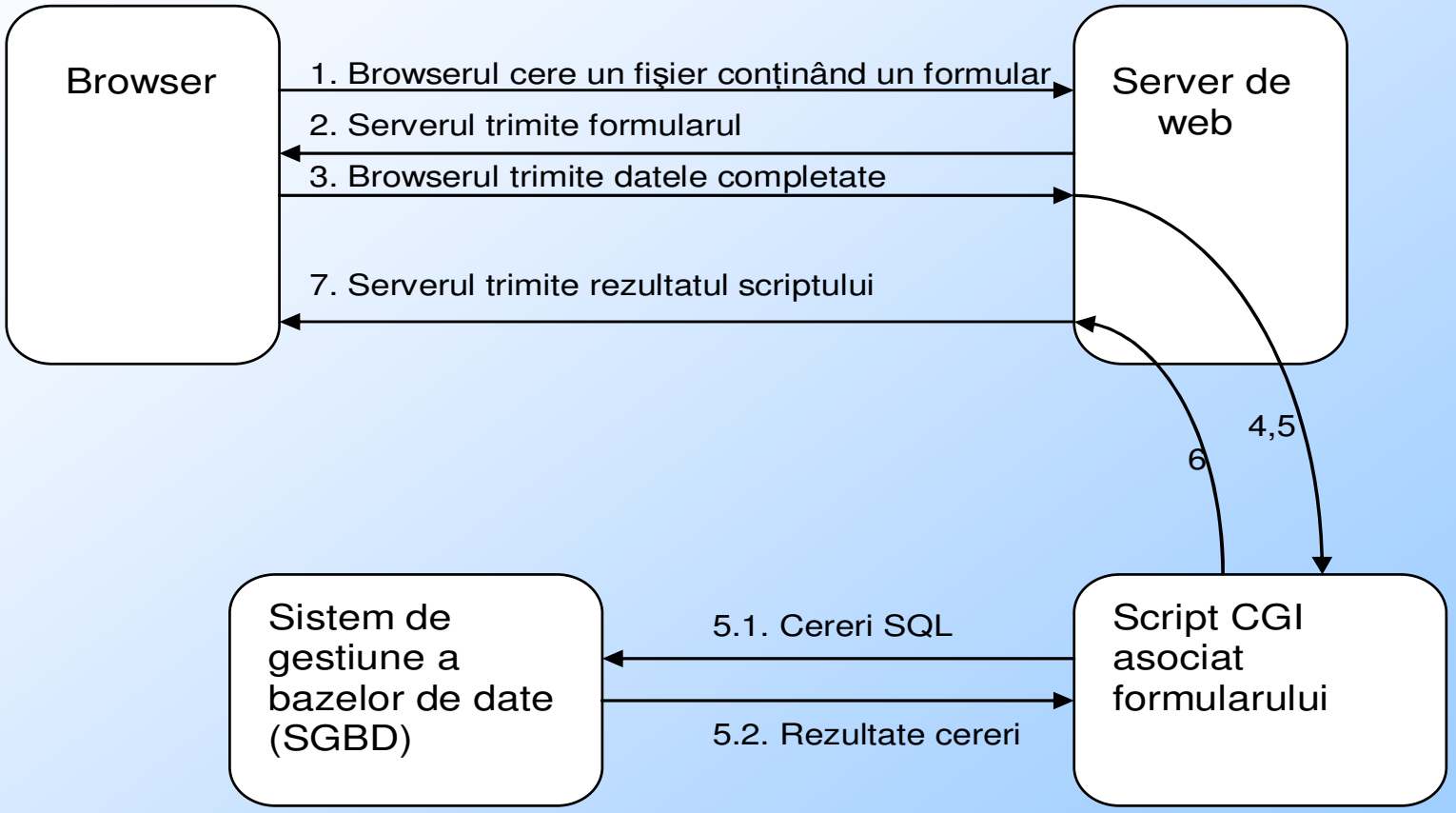
1. Browserul care poate fi oricare, ex.: Opera, Microsoft Internet Explorer, etc.
2. Serverul de web. Exista o multitudine de servere de web, printre care cel mai folosit in aplicatii de acest gen este Apache
3. Un sistem de gestiune a bazelor de date (SGBD) relaţional. Se poate folosi atat sistemul MySQL care practic gratuit dar si sisteme comerciale (Oracle, MS SQL Server, etc), acestea oferind in ultimul timp versiuni gratuite pentru aplicatii de dimensiuni mici.

Exemplu de arhitectura

4. Una sau mai multe formulare, care sunt fişiere HTML permitând utilizatorului introducerea prin tastare sau selectarea din meniuri sau prin butoane a unor date. Acestea vor defini anumite operaţii asupra bazei de date.
5. Scripturi CGI asociate formelor. Acestea sunt programe care interpretează datele completate în formular şi generează un raspuns către utilizator inclusiv prin interogarea unei baze de date.

- Fiecare dintre aceste componente poate exista sau rula pe o altă maşină.
- În mod obişnuit browserul rulează pe o maşină client iar celelalte pe una sau mai multe maşini aflate la furnizorul serviciului de web respectiv.

Arhitectura aplicatiei



Desfasurarea procesului

1. Browserul cere serverului de web o pagină HTML conţinând un formular.
2. Serverul trimite acea pagină browserului. Acesta o afişează pe ecran lăsând apoi utilizatorul să completeze elementele formularului.
3. După completare, utilizatorul selectează un buton de tip "trimite datele". Browserul împachetează informaţiile completate trimiţându-le serverului de web.
4. Serverul de web lansează scriptul CGI asociat formularului dându-i acestuia datele primite de la browser.

Desfasurarea procesului

5. Scriptul CGI despachetează aceste date şi le foloseşte ca informaţii de plecare pentru acţiunea pe care o efectuează. Printre operaţiile efectuate de acest script pot fi:
   5.1. Trimiterea de cereri SQL către un SGBD.
   5.2. Primirea şi interpretarea rezultatelor acestor cereri.
6. Scriptul CGI scrie la ieşire un fişier (în general de tip HTML dar nu numai) şi îşi încheie execuţia.
7. Serverul de web preia acest rezultat şi îl trimite browserului spre afişare.

Script CGI

- Scripturile CGI sunt o modalitate de a furniza pagini dinamice HTML sau de alt tip (imagini, video, sunet, postscript, etc).
- Acestea nu sunt efectiv stocate ca atare pe serverul web ci sunt generate pe baza datelor obţinute de la utilizator (în general conform datelor completate de către acesta într-un formular).
- Din această cauză ele nu pot fi servite direct de serverul web ci trebuiesc create de către un program care se numeşte script CGI.

Script CGI

- Numele de *script CGI* provine din denumirea specificaţiei interfeţei între serverul de web şi astfel de programe, numită ***C****ommon* ***G****ateway* ***I****nterface* - prescurtat CGI.
- Teoretic orice program executabil care respectă anumite convenţii în ceea ce priveşte mediul din care îşi colectează datele iniţiale şi modul în care scrie rezultatul poate fi un script CGI.

Exemplu

Cel mai simplu exemplu de script este un program care pentru orice date de intrare afişează un mesaj standard. Iată un exemplu scris în limbajul C:

```
#include <stdio.h>
int main()
{
printf("Content-Type: text/html\n\n");
printf("<html>\n");
printf("<body>\n");
printf("Sunt un script CGI\n");
printf("</body>\n);
printf("</html>\n");
}
```

Exemplu

- Rezultatul rulării acestui script este următorul:

```
Content-Type: text/html
    [Linie goala]
<html>
<body>
Sunt un script CGI
    </body>
</html>
```

Exemplu

- Observăm două porţiuni separate printr-o linie goală.
- Prima reprezintă un header conţinând tipul răspunsului (fişier de tip text în format HTML) iar a doua conţine efectiv fişierul respectiv.
- Mai multe informaţii despre sintaxa HTML se găsesc într-un capitol urmator.
- Detalierea modului în care scriptul îşi colectează informaţia de intrare şi a modului în care trebuie să arate rezultatul este făcută în paragrafele urmatoare.

Scripturi CGI

- Cele mai multe servere sunt deja preconfigurate pentru a putea lansa scripturi CGI. Câteva elemente comune sunt următoarele:
- Orice server web gestionează o anumită zonă din sistemul de fişiere al maşinii respective pornind dintr-un punct (director) numit SERVER_ROOT (pe serverele Apache/Unix acesta este de exemplu /usr/local/etc/httpd/). În general şi scripturile trebuiesc stocate în această zonă. Există cazuri în care se poate trece peste această restricţie dar o astfel de abordare poate crea probleme de securitate mărind posibilitatea accesului nepermis la datele stocate pe acea maşină.
- Unele servere web nu recunosc scripturile de tip program executabil decât dacă acestea au extensia .cgi.

Scripturi CGI

- Există configurări de servere în care scripturile trebuiesc stocate doar în directoare cu nume predefinit (de exemplu SERVER_ROOT/cgi-bin sau ~user/cgi-bin).
- Pe sistemele de tip Unix serverul de web rulează de obicei sub un user-Unix fară drepturi de login (de exemplu userul *nobody*). În acest caz calea de la SERVER_ROOT până la scriptul CGI trebuie să-i fie accesibilă: protecţie 701 pentru directoare şi 604 pentru scripturi sau +x pentru directoare şi +r pentru scripturi.
- Orice server web are unul sau mai multe fişiere de configurare (la Apache de exemplu fişierul se numeste httpd.conf) care sunt citite la lansare. Acestea conţin o serie de directive care afectează şi tratarea scripturilor CGI şi trebuiesc eventual modificate în anumite cazuri. După o astfel de modificare serverul trebuie relansat pentru a citi noile valori ale parametrilor.

Script CGI in pagina web

- Până acum am considerat un script CGI ca fiind un fişier executabil. Acesta poate proveni dintr-un program compilat (C sau alt limbaj) sau poate fi un fişier executabil scris în *shell* sau alt limbaj interpretat (cum este *perl*).
- În cazul multor servere web există însă şi posibilitatea de a include scripturile direct în fişierele HTML. În acest caz o pagină web va conţine atât directive HTML care sunt interpretate obişnuit de browser cât şi programe scrise în anumite limbaje care vor fi înlocuite cu rezultatul execuţiei lor. Există două abordări în acest caz:

Client side

- Abordarea client-side: paginile conţinând scripturi sunt trimise de serverul web fară nici o modificare către browser, urmând ca execuţia lor să aibă loc acolo.
- Este cazul limbajului *Java* (pentru acest caz programele incluse în paginile web sunt numite şi *applet*-uri) pe care browserele uzuale îl conţin nativ.
- În felul acesta o parte a procesărilor se transferă de la server micsorând volumul de activitate pentru primul.

Server side

- Abordarea server-side: serverul înlocuieşte în pagina cerută programul cu rezultatul acestuia.
- Pentru aceasta serverul trebuie să aibă suport pentru limbajul respectiv.
- In prima parte a cursului nostru vom prezenta limbajul de scripting server side PHP

CGI – date de intrare

- Fiecare formular are asociat o metodă prin care datele completate de utilizator sunt transmise de serverul de web către scriptul CGI.
- Aceste informaţii sunt împachetate sub forma unui şir de perechi de forma:

```
simbol1=valoare1&simbol2=valoare2&…&simbolk=valoarek
```

- unde numele simbolilor este dat de numele câmpurilor formularului iar valorile sunt cele furnizate de utilizator prin completare.

CGI – date de intrare

- Metodele care pot fi folosite sunt în număr de două şi anume:

1. Metoda GET pentru care datele sunt stocate într-o variabilă de mediu (*environment*) cu nume predefinit (QUERY_STRING).
2. Metoda POST pentru care datele sunt plasate în fişierul standard de intrare al scriptului (*stdin* în cazul scripturilor scrise în limbajul C) iar lungimea şirului care le conţine este stocată într-o variabilă cu nume predefinit (CONTENT_LENGTH).

CGI – date de intrare

- Pe lângă aceste informaţii serverul setează o serie de alte variabile de mediu conţinând o multitudine de alte informaţii.
- În cazul scrierii de scripturi în limbajul C acestea se pot obţine folosind funcţia *getenv()* iar în cazul scripturilor care utilizează modulul PHP acestea sunt direct disponibile.
- Informaţiile furnizate cuprind:

Variabile de mediu

- Informaţii generale: Versiunea interfetei CGI utilizate, protocolul folosit şi metoda utilizată de formular: GET sau POST
- Datele primite de la browser conţinând informaţiile furnizate de utilizator prin intermediul completării câmpurilor formularului.
- Informaţii despre client: numele şi adresa maşinii client, numele userului şi modul de autentificare dacă e cazul, tipul browserului folosit, limbile suportate de acesta, tipurile de date acceptate, pagina precedentă accesată, etc.

Variabile de mediu

- Informaţii despre serverul de web care lansează scriptul: tipul serverului, numele sau, portul pe care a fost primită cererea, rădăcina arborelui de documente (DOCUMENT_ ROOT) precum şi numele şi calea către scriptul lansat.
- Alte informaţii primite de la client: căi suplimentare către alte documente şi translaţia lor în contextul sistemului de fişiere al maşinii pe care se află.

Variabile de sistem

- Pe baza variabilelor de mediu, a datelor primite, etc, mediul PHP prseteaza pentru executia scriptului o serie de variabile care pot fi folosite apoi direct de catre acesta.
- Marea lor majoritate sunt tablouri asociative, elementele tablourilor respective putand fi accesate prin numele lor si nu doar prin indici ca in cazul tablourilor obisnuite.
- Cateva dintre cele mai importante astfel de variabile sunt:
  - $_GET, $_POST – tablouri asociative care contin informatiile trimise de formular
  - $_SESSION – tablou asociativ care permite transmiterea de valori de la o pagina PHP la alta.

CGI - Date de iesire

- Un script CGI trebuie să scrie la ieşirea standard un rezultat cu următoarea formă:

1. Un header format dintr-o succesiune de directive, câte una pe linie. Acesta va conţine opţional directiva Status iar în caz de succes una din cele două directive Content-type sau Location (dar nu ambele).
2. O linie goală.
3. Un document de tipul specificat în Content-type.

Status

Directiva Status:

- Această directivă are sintaxa:

Status: <cod_numeric> [<denumire>]

- Codurile numerice uzuale sunt date în continuare (vezi si [3])
- Directiva se poate folosi în cazurile în care se doreşte semnalarea unei erori.
- Ea va fi transmisă mai departe de serverul de web către browser care va afişa un mesaj de eroare corespunzător (un text explicativ se poate include în zona de document a răspunsului, după linia goală care marchează "sfârşit de header".
- În caz de succes se poate folosi cu codul 200

Status

Cod	***Descriere***
200	Succes. Răspunsul corespunzător urmează după header.
201	Dacă o resursa (fişier de exemplu) a fost creată de către server se poate trimite acest cod împreună cu locația acesteia. Se foloseşte în cazul metodei POST pentru "file uploading".
202	Cerere acceptată dar încă neprocesată. Se poate folosi atunci când rezultatul va fi trimis ulterior (de exemplu prin email).
204	Succes dar răspunsul corespunzător este vid.
301	Documentul a fost mutat permanent la altă adresă. Aceasta se specifică cu Location.
302	Documentul a fost mutat temporar la altă adresă. Aceasta se specifică cu Location.
304	Documentul nu este servit. Se poate utiliza atunci când cererea a fost condițională (de exemplu documentul solicitat a fost modificat după o anumită dată) iar condiția nu este îndeplinită.
400	Cerere eronată. În general acest cod nu poate fi primit decât dacă browserul conține erori.
401	Cererea priveşte un document care necesită autentificarea utilizatorului.
403	Cerere corectă dar care nu poate fi servită (de exemplu serverul sau clientul nu au drepturile necesare de acces).
404	Fişierul cerut nu există.
500	Eroare server. Acest cod este trimis dacă scriptul CGI are erori sau nu trimite headerele necesare.
501	Tip de cerere neimplementată pe acel server
502	Este trimisă de server atunci când el acționează ca "proxy server" sau "gateway" şi a primit un raspuns eronat de la un alt server.
503	Serverul este prea solicitat pentru a trata alte cereri.

Content type

Directiva Content-type

- Această directivă este esenţială pentru ca serverul de web să ştie ce fel de document urmează.
- Forma ei este:

```
Content-type: tip/subtip
```

- Cateva tipuri uzuale care se pot include în această directivă sunt cele din tabelul urmator.
- Valoarea implicită este *text/plain*, însa este recomandat ca ea să nu lipsească deoarece serverul de web poate să nu înţeleagă rezultatul scriptului şi să transmită o eroare de tip 500 (*Internal server error*) către browser.

Content Type

Tip/Subtip	***Descriere***
Text/plain	Text. Este tipul implicit.
Text/html	Fişier HTML.
Text/richtext	Rich Text Format (RTF). Este un tip recunoscut de majoritatea procesoarelor de texte.
Image/gif	Imagine în format GIF, recunoscută de majoritatea browserelor.
Image/jpeg	Imagine în format JPEG, de asemenea recunoscută de browsere.
Image/x-xbitmap	Imagine în format X bitmap, un format foarte simplu recunoscut de browsere. Extensia standard pentru astfel de fişiere este .xbm
audio/basic	Sunet în format ulaw comprimat. Extensia standard pentru astfel de fişiere este .au.
audio/x-wav	Sunet în format specific Microsoft Windows.
video/mpeg	Imagini video în format MPEG comprimat.
video/quicktime	Imagini video în format Quicktime video.
video/x-msvideo	Imagini video în format Microsoft Video. Extensia uzuală este .avi.
application/octet-stream	Fişier binar. La primirea unui astfel de fişier browserul permite pe baza dialogului cu utilizatorul salvarea pe discul local.
application/postscript	Fişiere Postscript.

Location

Directiva Location

- Forma acestei directive este:

```
Location: URL
```

- Prin această directivă se semnalează că rezultatul trebuie căutat la adresa specificată.
- Browserul va fi automat redirectat şi va prezenta documentul respectiv.
- Din acest motiv în cazul folosirii directivei *Location* după linia goală care marchează "Sfarşit de header" răspunsul poate să nu conţină altceva.

Erori returnate de server

Eroarea 403: Forbidden (Acces interzis)

- În mediul Unix fiecare user are sau nu drepturi de citire (*read*- r) scriere (*write* - w) sau execuţie (*execute* - x) asupra directoarelor şi fişierelor.
- Cum serverul web rulează de obicei sub userul nobody trebuie ca întreaga cale de la SERVER_ROOT până la scriptul CGI să fie accesibilă acestuia.
- Pentru aceasta directoarele trebuie să permită dreptul de execuţie pentru server iar fişierele conţinând scripturi cgi şi forme drepturi de citire şi execuţie. În caz contrar se va semnala această eroare.

Erori returnate de server

Eroarea 404: File not found (fişier inexistent)

- Această eroare apare atunci când referinţa absolută sau relativă la un fişier conţinând un formular sau un script CGI este greşită.
- Este des întâlnită atunci când în formular se specifica o adresă relativă a scriptului asociat.
- În cazul mutării formularului în alt director referinţa nu mai este valabilă.

Erori returnate de server

Eroarea 500: Internal Server Error (Eroare internă server)

- Această eroare apare atunci când serverul web nu poate interpreta rezultatul scriptului. Aşa cum am văzut, un rezultat corect constă în:

1. Un header care conţine opţional directiva Status iar în caz de succes una din cele două directive Content-type sau Location.
2. O linie goală.
3. Corpul documentului, compatibil cu Content-type dacă aceasta este prezentă în header

- În cazul în care rezultatul scriptului nu arată astfel serverul va semnala această eroare.

Erori returnate de server

Eroarea 501: Method not implemented (metodă neimplementată)

- Această eroare poate apare în cazul metodei POST dacă serverul web a fost configurat să nu permită execuţia scripturilor în anumite directoare.
- Toate scripturile stocate în acestea vor fi tratate ca fişiere obişnuite pentru care metoda POST este interzisă.

Alte situatii

Directivele de header ale răspunsului (Content-type de exemplu) apar pe ecranul browserului.

- Această situaţie apare când directivele de header ale răspunsului au fost emise de scriptul CGI mai târziu decât trebuie, ele fiind interpretate ca făcând parte din documentul propriu-zis.

Alte situatii

În locul unui rezultat se obţine un fişier binar indescifrabil

- În acest caz este posibil ca cererea să fi fost făcută fără a se trece prin serverul de web.
- Această situaţie poate apare atunci cand browserul şi formularul sunt pe aceeaşi maşină iar accesul la formular a fost făcut direct (majoritatea browserelor pot fi folosite şi pentru a afişa fişiere HTML fără intervenţia serverului de web).

Alte situatii

Modificările făcute în formulare sau scripturi nu se reflectă în răspunsul primit de la server.

- Multe servere (inclusiv Apache) se pot configura cu *caching*.
- Acest lucru înseamnă că serverul stochează în cache ultimele pagini servite către clienţi iar la o nouă cerere pentru una dintre acestea nu mai citeşte pagina de la locaţia originala ci din *cache*.
- Pentru a forţa reîncărcarea paginii se poate folosi butonul *Reload* existent la majoritatea browserelor.

Bibliografie

1. Vispi Munshi, Developing a technical architecture for Web-based enterprise software systems, http://www.ibm.com/developerworks/web/library/wa-techarch/
2. F. Radulescu, Baze de date in Internet, Editura Printech, 2000
3. Hypertext Transfer Protocol -- HTTP/1.1, Cap. 10 Status Code Definitions, http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Sfârşitul Capitolului 1