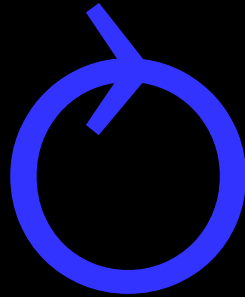


# Tehnologii Web

## programare Web (I)



protocolul HTTP  
*cookie*-uri și sesiuni

“Orice intrare trebuie plătită.”

**Mihai Giugariu**

# Ce este Web-ul?

**Web**-ul = spațiu informațional compus  
din elemente de interes, numite **resurse**,  
desemnate de identificatori globali – **URI/IRI**

**Web**-ul = spațiu informațional compus  
din elemente de interes, numite **resurse**,  
desemnate de identificatori globali – **URI/IRI**

detalii la [www.w3.org/TR/webarch/](http://www.w3.org/TR/webarch/)  
recomandare W3C, 2004

# resurse Web

## Aspecte de interes

identificarea

interacțiunea

reprezentarea prin formate de date

# resurse Web

## Aspecte de interes

protocol:  
HTTP

identificarea

URI/IRI

interacțiunea

reprezentarea prin formate de date

limbaj(e)  
de marcare

Cum are loc interacțiunea  
dintre client(i) și server(e) Web?



# HTTP

*HyperText Transfer Protocol*

bazat pe TCP/IP

<http://www.w3.org/Protocols/>

# HTTP

situat la nivelul aplicație

transfer de hipertext/hipermedia  
(HTTP – *HyperText Transfer Protocol*)

transport fiabil via *socket*-uri  
(TCP – *Transmission Control Protocol*)

interconectare & dirijare  
(IP – *Internet Protocol*)

controlul accesului la mediul de transmitere  
a datelor (MAC – *Medium Access Control*)

# HTTP

## *HyperText Transfer Protocol*

protocol fiabil, de tip cerere/răspuns

port standard de acces: **80**

# HTTP

## *HyperText Transfer Protocol*

HTTP/0.9, HTTP/1.0  
în trecut

**HTTP/1.1** standardizat în 1999: RFC 2616  
din iunie 2014, e definit de RFC 7230—7235

**HTTP/2**

în viitorul apropiat – <http://http2.github.io/>

# HTTP: arhitectura

## Server Web

*daemon* – „spirit protector”

## Client Web

navigator (*browser*), robot (*crawler*), *player*,...

# HTTP: arhitectura

## Server Web

Apache, Internet Information Services, Lighttpd, Nginx,...

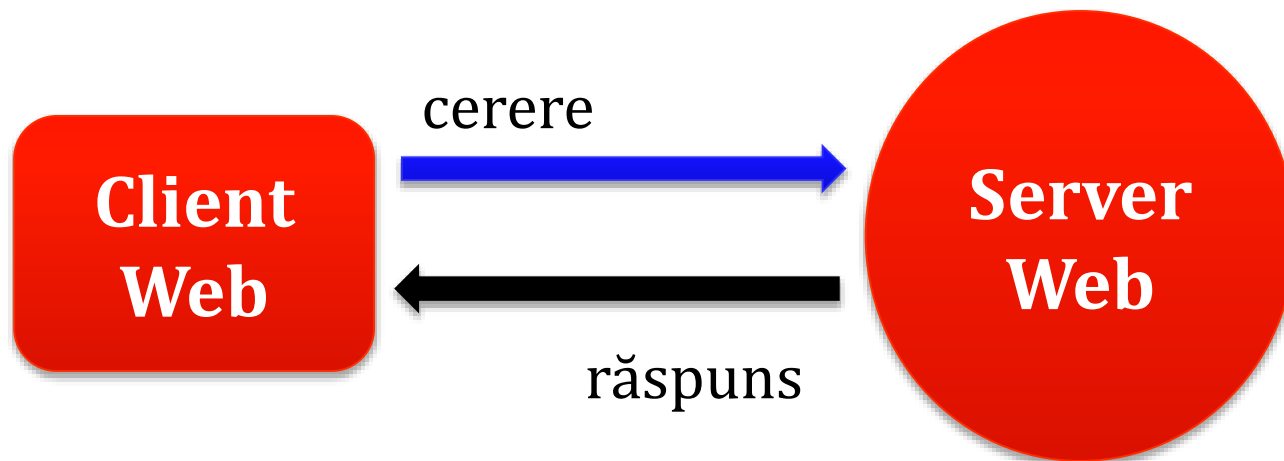
## Client Web

Mosaic ► Netscape ► Mozilla ► Firefox,  
Internet Explorer, Chromium, wget, iTunes, Echofon etc.  
detalii în prezentarea „Arhitectura navigatorului Web”:  
[www.slideshare.net/busaco/web02-arhitectura-browserweb-43733189](http://www.slideshare.net/busaco/web02-arhitectura-browserweb-43733189)

# HTTP

## Cererea și răspunsul

accesarea – eventual, modificarea – reprezentării  
resursei via URI-ul asociat



# HTTP: termeni

## Mesaj

unitatea de bază a unei comunicații HTTP  
(cerere sau răspuns)



# HTTP: termeni

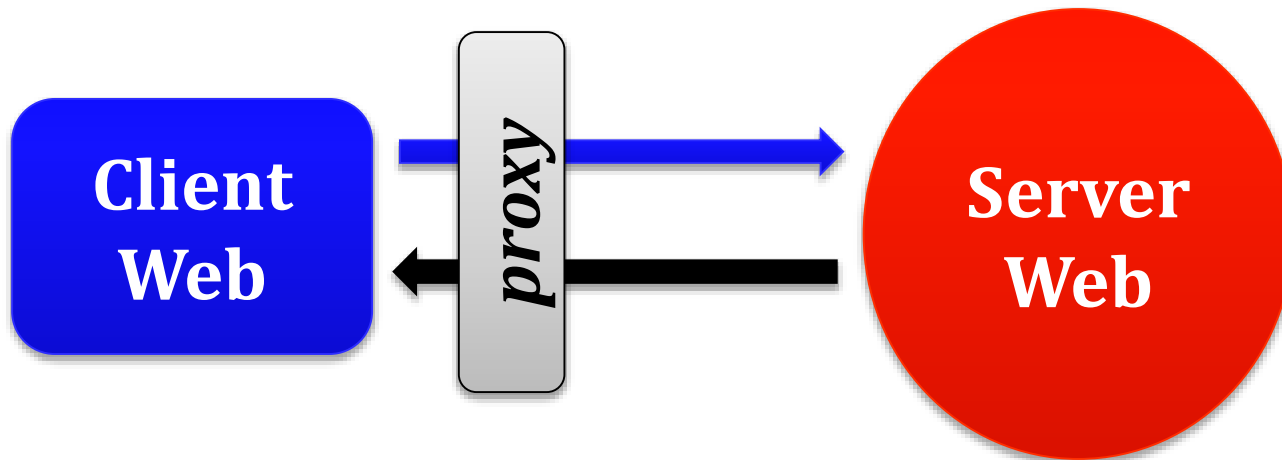
## Intermediar

*proxy*  
poartă  
tunel

# HTTP: termeni

## *Proxy*

localizat în proximitatea clientului/serverului  
are rol atât de server, cât și de client



# HTTP: termeni

## *Proxy*

### *forward proxy*

intermediar pentru clienții din vecinătate

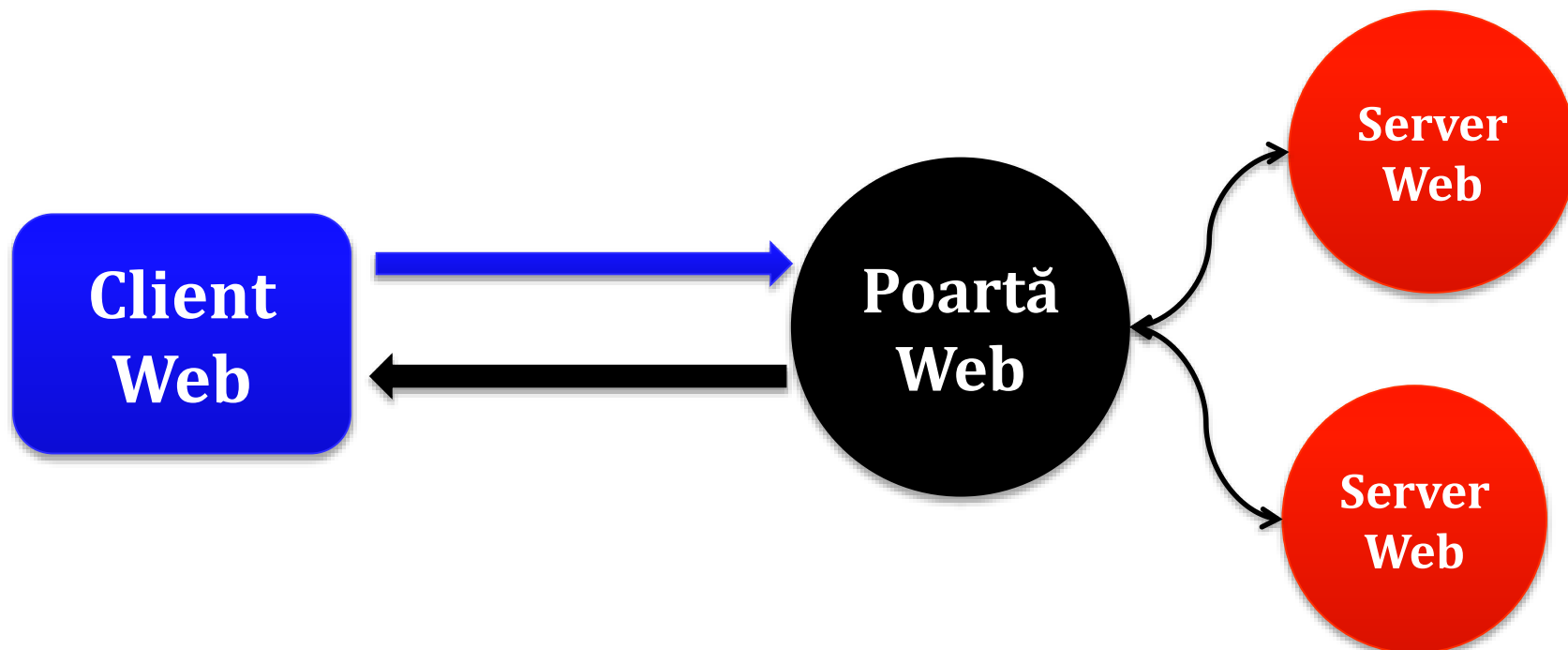
### *reverse proxy*

intermediar pentru serverele din vecinătate

# HTTP: termeni

## Poartă (*gateway*)

intermediar care ascunde serverul țintă,  
clientul neștiind aceasta



# HTTP: termeni

## Tunel

rol de retransmitere – eventual, criptată –  
a mesajului

# HTTP: termeni

## Tunel

rol de retransmitere – eventual, criptată –  
a mesajului

context: protocolul **HTTPS** – asigură comunicații „sigure”  
HTTP via TLS (*Transport Layer Security*):  
autentificare pe baza certificatelor digitale  
+ criptare bidirecțională

# HTTP: termeni

## *Cache*

zonă locală de stocare – în memorie, pe disc –  
a mesajelor (datelor) la nivel de server/client

# HTTP: termeni

## *Cache*

zonă locală de stocare – în memorie, pe disc – a mesajelor (datelor) la nivel de server/client

context: asigurarea performanței aplicațiilor Web



# HTTP: **mensaje**

Mensaje HTTP = **header + body**

# HTTP: mesaje

## Antet

include o mulțime de câmpuri

**Message-header ::= field-name ":" [ field-value ] CRLF**

# HTTP: mensaje

## Cerere HTTP

**Request** ::= **Method Request-URI** **ProtocolVersion** **CRLF**  
[ **Message-header** ] [ **CRLF MIME-data** ]

**GET** /~busaco/teach/courses/web/ **HTTP/1.1** **CRLF**  
**Host: profs.info.uaic.ro**

# HTTP: mesaje

## Răspuns HTTP

Status-line ::= HTTP-vers Digit Digit Digit Reason  
CRLF Content

HTTP/1.1 200 OK CRLF ...

# HTTP: metode

## GET

cerere – efectuată de un client – pentru accesul la reprezentarea unei resurse

document HTML, foaie de stiluri CSS,  
image în format PNG, ilustrație vectorială SVG,  
program JavaScript, flux de știri Atom (XML),  
prezentare PDF, date în format JSON,...

# HTTP: metode

## HEAD

similară cu GET, dar în mod uzual  
se doresc doar meta-date

*e.g.*, tipul MIME al resursei, ultima actualizare,...

# HTTP: metode

## PUT

actualizează o reprezentare de resursă sau  
eventual creează o resursă la nivel de server Web

# HTTP: metode

## POST

creează o resursă, trimittând uzual entittăți  
(date, acțiuni) spre server

*e.g.*, datele dintr-un formular Web



# HTTP: metode

## DELETE

șterge o resursă – reprezentarea ei – de pe server

# HTTP: metode

## Remarcă

uzual, *browser*-ul Web permite doar folosirea metodelor GET și POST

# HTTP: metode

O metodă e considerată *sigură* (*safe*)  
dacă nu conduce la modificarea stării serverului

GET și HEAD sunt *safe*

POST, PUT și DELETE nu sunt *safe*

# HTTP: metode

O metoda e considerată **idempotentă** în cazul în care cereri identice vor conduce la returnarea aceluiasi răspuns (aceeași reprezentare)

GET, PUT și DELETE sunt idempotente  
POST nu este idempotentă

# HTTP: reprezentări ale resursei

Codificarea setului de caractere (*encoding*)

ISO-8859-1

ISO-8859-2

KOI8-R

ISO-2022-JP

UTF-8

UTF-16 Little Endian

...

# HTTP: reprezentări ale resursei

## Codificarea mesajelor

comprimare, asigurarea identității  
și/sau integrității

uzual: `gzip`

# HTTP: reprezentări ale resursei

## Formatul reprezentării

text

HTML, CSS, text obișnuit, cod JavaScript, document XML

sau

binar

imagini (JPEG, PNG), documente PDF, resurse multimedia

# HTTP: reprezentări ale resursei

Tipul conținutului resursei

*MIME type*

<http://www.iana.org/assignments/media-types/media-types.xhtml>



# HTTP: câmpuri (attribute)

## Content-Type

permite transferul datelor de orice tip

**Content-Type:** tip/subtip

# HTTP: câmpuri (attribute)

## Content-Type

specificat prin **MIME**  
(*Multipurpose Internet Mail Extensions*)

desemnează un set de **tipuri primare de conținut**  
+ **sub-tipuri** adiționale

inițial, utilizat în contextul poștei electronice

# HTTP: câmpuri (attribute)

## Tipuri MIME principale

**text** desemnează formate textuale

**text/plain** – fișier text neformatat

**text/html** – document HTML

**text/css** – foaie de stiluri CSS

# HTTP: câmpuri (attribute)

Tipuri MIME principale

**image** specifică formate grafice

**image/gif** – imagini GIF (*Graphics Interchange Format*)

**image/jpeg** – fotografii JPEG (*Joint Picture Experts Group*)

**image/png** – imagini PNG (*Portable Network Graphics*)

# HTTP: câmpuri (attribute)

## Tipuri MIME principale

**audio** desemnează conținuturi sonore

**audio/mpeg** – resursă codificată în format MP3  
specificația privitoare la date audio a standardului MPEG  
(*Motion Picture Experts Group*)

**audio/ac3** – resursă compresată conform standardului AC-3

# HTTP: câmpuri (attribute)

## Tipuri MIME principale

**video** definește conținuturi video:  
animații, filme

**video/h264** – resursă în format H.264

**video/ogg** – conținut codificat în formatul deschis OGG

# HTTP: câmpuri (attribute)

## Tipuri MIME principale

**application** desemnează formate care vor putea fi procesate de aplicații disponibile la nivel de client

**application/javascript** – program JavaScript

**application/json** – date JSON (*JavaScript Object Notation*)

**application/octet-stream** – șir arbitrar de octeți


# HTTP: câmpuri (attribute)

Tipuri MIME principale

**multipart** utilizat la transferul datelor compuse

**multipart/mixed** – conținut mixt

**multipart/alternative** – conținuturi alternative



*e.g., calități diferite de  
stream-uri multimedia*



# HTTP: câmpuri (attribute)

## Location

**Location** ":" "http://" **authority** [ ":" **port** ] [ **abs\_path** ]

redirecționează clientul spre o altă reprezentare a resursei  
(*HTTP redirect*)

# HTTP: câmpuri (attribute)

## Location

**Location** ":" "http://" authority [ ":" port ] [ abs\_path ]

**Location:** http://www.infoiasi.ro:8080/s-a\_mutat.html

# HTTP: câmpuri (attribute)

## Referer

desemnează URI-ul resursei Web  
care a referit resursa curentă

folosit pentru a determina de unde provin  
accesările unui document dat

# HTTP: câmpuri (attribute)

## Host

specifică adresa – IP sau simbolică – a mașinii de pe care se solicită accesul la o resursă

# HTTP: câmpuri (attribute)

Sunt definite și altele, vizând:

conținutul acceptat (*content negotiation*) – *e.g.*, Accept  
autentificare & autorizare – WWW-Authenticate Authorization  
acces condiționat la resurse – If-Match, If-Modified-Since, ...  
*cache-ul* – Cache-Control, Expires, ETag etc.  
*proxy-ul* – Proxy-Authenticate, Proxy-Authorization, Via  
...și altele

# HTTP: starea

## Coduri de informare (1xx)

100 Continue

101 Switching Protocols

# HTTP: starea

## Coduri de **succes** (2xx)

200 Ok

201 Created

202 Accepted

204 No Content

206 Partial Content

# HTTP: starea

## Coduri de **redirectare** (3xx)

300 Multiple Choices

301 Moved Permanently

303 See Other

304 Not Modified

305 Use Proxy



# HTTP: starea

## Coduri de eroare la nivel de client (4xx)

400 Bad Request

403 Forbidden

404 Not Found

405 Method Not Allowed

408 Request Timeout

# HTTP: starea

## Coduri de eroare la nivel de server (5xx)

500 Internal Server Error

501 Not Implemented

502 Bad Gateway

503 Service Unavailable

504 Gateway Timeout

# HTTP: jurnalizare

Cererile adresare serverului Web sunt jurnalizate

*Common Log Format*

format standardizat

pentru Apache, vezi modulul **mod\_log\_config**

c12.uaic.ro - msi2013 [13/Feb/2014:14:53:14 +0200]  
"GET /~vidrascu/MasterSI2/note/Restanta.pdf HTTP/1.1" 206 25227  
"http://profs.info.uaic.ro/~vidrascu/MasterSI2/index.html" "...Firefox/27.0"  
82-137-8-231.rdsnet.ro - - [13/Feb/2014:15:38:23 +0200]  
"POST /~computernetworks/login.php HTTP/1.1" 302 1115  
"http://profs.info.uaic.ro/~computernetworks/login.php"  
"Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:26.0) Gecko/20100101 Firefox/26.0"  
ec2-23-21-0-202.compute-1.amazonaws.com - - [13/Feb/2014:15:48:29 +0200]  
"GET /~busaco/teach/courses/web/presentations/web01ArhitecturaWeb.pdf HTTP/1.1"  
200 2081804 "-" "HTTP\_Request2/2.2.0 (http://pear.php.net/package/http\_request2)..."  
199.16.156.126 - - [13/Feb/2014:15:58:58 +0200]  
"GET /robots.txt HTTP/1.1" 404 182 "-" "Twitterbot/1.0"  
psihologie-c-113.psih.uaic.ro - - [13/Feb/2014:16:03:04 +0200]  
"GET /~busaco/ HTTP/1.1" 200 1942 "-" "Mozilla/5.0 (X11; Linux x86\_64; ...) Firefox/27.0"  
psihologie-c-113.psih.uaic.ro - - [13/Feb/2014:16:03:04 +0200]  
"GET /~busaco/csb.css HTTP/1.1" 200 852 "http://profs.info.uaic.ro/~busaco/"  
"Mozilla/5.0 (X11; Linux x86\_64; rv:27.0) Gecko/20100101 Firefox/27.0"  
proxy-220-255-2-224.singnet.com.sg - - [13/Feb/2014:16:23:23 +0200]  
"GET /favicon.ico HTTP/1.1" 200 1406 "-" "Dalvik/1.6.0 (Linux; U; Android 4.0.4; ...)"  
c2.uaic.ro - - [13/Feb/2014:16:33:43 +0200]  
"GET /~busaco/teach/courses/web/ HTTP/1.1" 304 - "-" "... Chrome/32.0.1700.107..."  
220.181.51.219 - - [13/Feb/2014:19:20:20 +0200]  
"HEAD /%7Ebusaco/music/09.Sabin%20Buraga%20-...mp3 HTTP/1.0" 200 - "-"  
"NSPlayer/10.0.0.4072 WMFSDK/10.0"

**GET** /~busaco/teach/courses/web/web-film.html **HTTP/1.1**  
**Host:** [profs.info.uaic.ro](http://profs.info.uaic.ro)  
**User-Agent:** Mozilla/5.0 (iPad; CPU OS 8\_1\_3 like Mac OS X)  
AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0  
Mobile/12B466 Safari/600.1.4  
**Accept:** [text/html](#),[application/xhtml+xml](#);q=0.9,[\\*/\\*](#);q=0.8  
**Accept-Language:** en-us,en;q=0.5  
**Accept-Encoding:** gzip, deflate  
**Connection:** keep-alive  
**Referer:** <http://profs.info.uaic.ro/~busaco/teach/courses/web/>

**HTTP: exemplu de cerere**

**HTTP/1.1 200 OK**

**Date:** Mon, 23 Feb 2015 15:18:01 GMT

**Server:** Apache

**Last-Modified:** Mon, 23 Feb 2015 07:46:02 GMT

**Content-Encoding:** gzip

**Content-Length:** 1498

**Keep-Alive:** timeout=15, max=100

**Connection:** Keep-Alive

**Content-Type:** text/html

câmpuri-antet  
(meta-date)

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"  
lang="ro" xml:lang="ro">

...

</html>

**HTTP: exemplu de răspuns**

## Destination

Follow redirects: [Off](#)

## Authentication

## Headers

## Parameters

[remove all](#)

GET

GET  
POST  
PUT  
PATCH  
HEAD  
OPTIONS  
DELETE  
[+ Add Header\(s\)](#)

tags

[Add another parameter](#)

For GET, HEAD and OPTIONS requests, parameters will be added to the querystring in the requested URL.

[⚡ Launch Request](#)

inspectarea online a mesajelor HTTP  
via [www.hurl.it](http://www.hurl.it)

GET https://api.flickr.com/services/feeds/photos\_public.gne?tags=lasi,Fil

200 OK 46.46 kB 2232 ms

[🔄 Edit and Retry](#)[View Request](#)[View Response](#)

## HEADERS

Accept: \*/\*  
Accept-Encoding: gzip, deflate, compress  
User-Agent: runscope/0.1

## QUERYSTRING

tags: lasi,Fil

GET https://api.flickr.com/services/feeds/photos\_public.gne?tags=iasi,FII

200 OK 46.46 kB 2232 ms

[View Request](#)

[View Response](#)

## HEADERS

**Age:** 6

**Cache-Control:** no-store, no-cache, must-revalidate, private, post-check=0, pre-check=0

**Connection:** keep-alive

**Content-Type:** application/atom+xml; charset=utf-8

**Date:** Wed, 19 Feb 2014 11:00:04 GMT

**Expires:** Mon, 26 Jul 1997 05:00:00 GMT

**Last-Modified:** Mon, 31 May 2010 11:56:03 GMT

**P3p:** policyref="http://info.yahoo.com/w3c/p3p.xml", CP="CAO DSP COR CUR ADM DEV TAI PSA PSD IVAi IVDi CONi TELo OTPi OUR DELi SAMi OTRi UNRi PUBi IND PHY ONL UNI PUR FIN COM NAV INT DEM CNT STA POL HEA PRE LOC GOV"

**Pragma:** no-cache

**Server:** ATS

**Set-Cookie:** xb=572247; expires=Sat, 20-Feb-2016 11:00:06 GMT; path=/; domain=.flickr.com

**Transfer-Encoding:** chunked

**Via:** http/1.1 fts123.flickr.bf1.yahoo.com (ApacheTrafficServer/4.0.1 [cMsSf ]), http/1.1 r10.ycpi.ac4.yahoo.net (ApacheTrafficServer/4.0.2 [cMsSf ])

**X-Served-By:** www55.flickr.bf1.yahoo.com

expiră în trecut  
(nu va fi păstrat în *cache*)

câmpurile X- nu  
sunt standardizate

## BODY

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
```

```
<feed
```

```
  xmlns="http://www.w3.org/2005/Atom"
```

```
  xmlns:dc="http://purl.org/dc/elements/1.1/"
```

```
  xmlns:flickr="urn:flickr:user"
```

```
  xmlns:media="http://search.yahoo.com/mrss/"
```

```
  <title>Recent Uploads tagged iasi and fii</title>
```

```
  <link rel="self" href="http://api.flickr.com/services/feeds/photos_public.gne?tags=Iasi%2CFII" />
```

```
  <link rel="alternate" type="text/html" href="http://www.flickr.com/photos/" />
```

conținut propriu-zis  
(flux Atom)  
procesat de client

[view raw](#)



# HTTP: API-uri

**cURL + libcurl**

(C, Java, Haskell, .NET, PHP, Ruby,...) – <http://curl.haxx.se/>

**Apache HttpComponents** (Java) – <http://hc.apache.org/>

**httplib** (Python 2) + **http.client** (Python 3)

**neon** (bibliotecă C): <http://www.webdav.org/neon/>

**WinHTTP**

(specific Windows: C/C++) – <http://tinyurl.com/6eemqqc>

# HTTP: instrumente

extensia **Firebug** pentru Firefox  
(la nivel de client; JavaScript) – <http://getfirebug.com/>

**Google Chrome Developer Tools**  
<http://code.google.com/chrome/devtools>

**Fiddler** (*free Web debugging proxy*):  
[www.telerik.com/fiddler](http://www.telerik.com/fiddler)

# Care e arhitectura serverului Web?

# HTTP: server Web

Deservește cereri multiple provenite de la clienți  
pe baza protocolului HTTP

# HTTP: server Web

Deservește cereri multiple provenite de la clienți  
pe baza protocolului HTTP

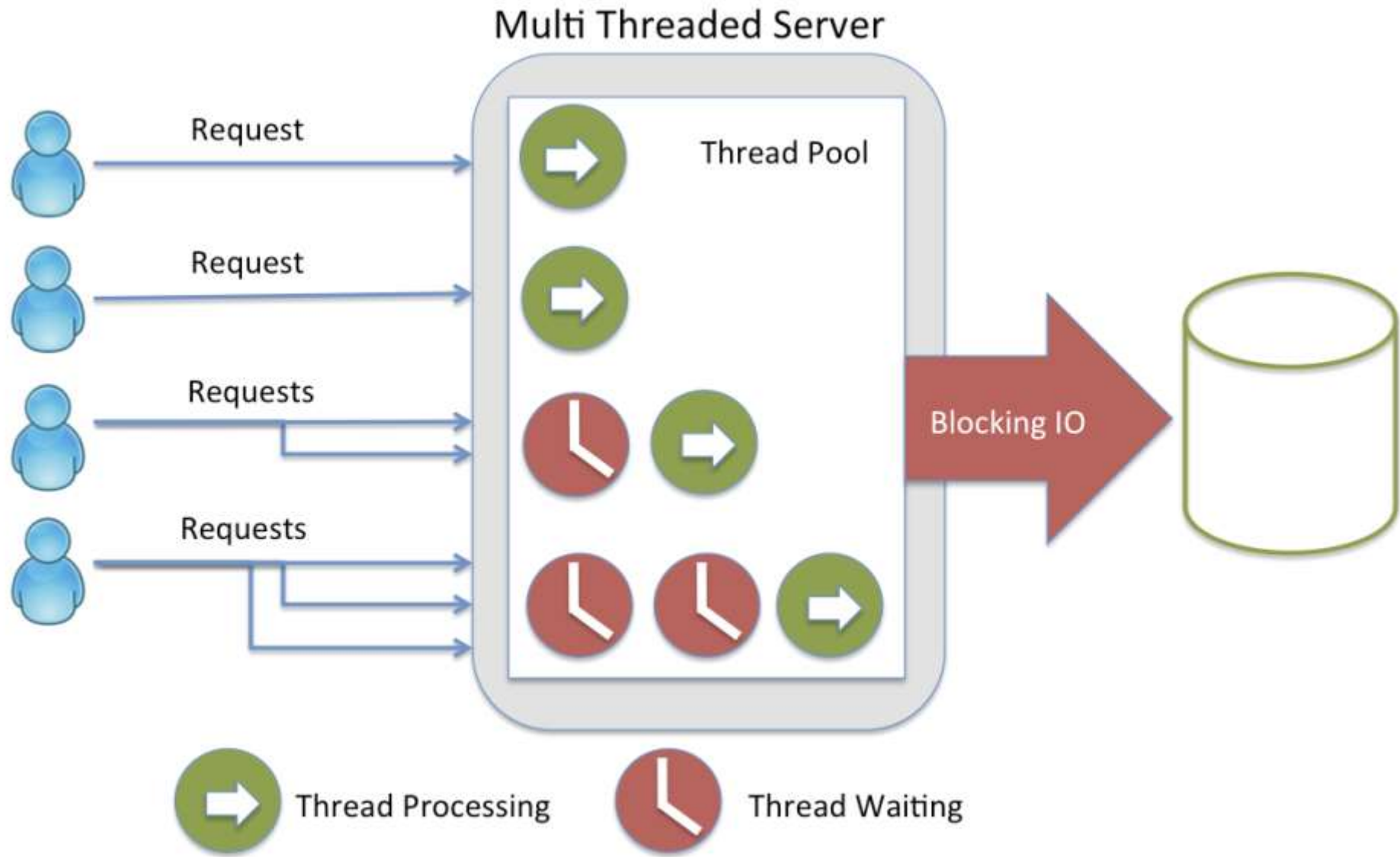
fiecare cerere e considerată independentă de alta,  
chiar dacă provine de la același client Web

- nu e păstrată starea conexiunii – *stateless*

# HTTP: server Web

Tradițional, implementarea serverului Web este una *pre-forked* sau *pre-threaded*

se creează un număr de procese copil ori fire de execuție (*threads*) la inițializare, fiecare proces/fir interacționând cu un anumit client



# HTTP: server Web

Comportamentul serverului poate fi stabilit  
via diverși parametri (directive) de configurare



# HTTP: server Web

Studiu de caz: configurarea serverului Apache  
(din aprilie 1996, cel mai utilizat server Web)

<http://httpd.apache.org/>

configurația generală prin fișierul **httpd.conf**  
implicit se creează 6 instanțe **httpd**

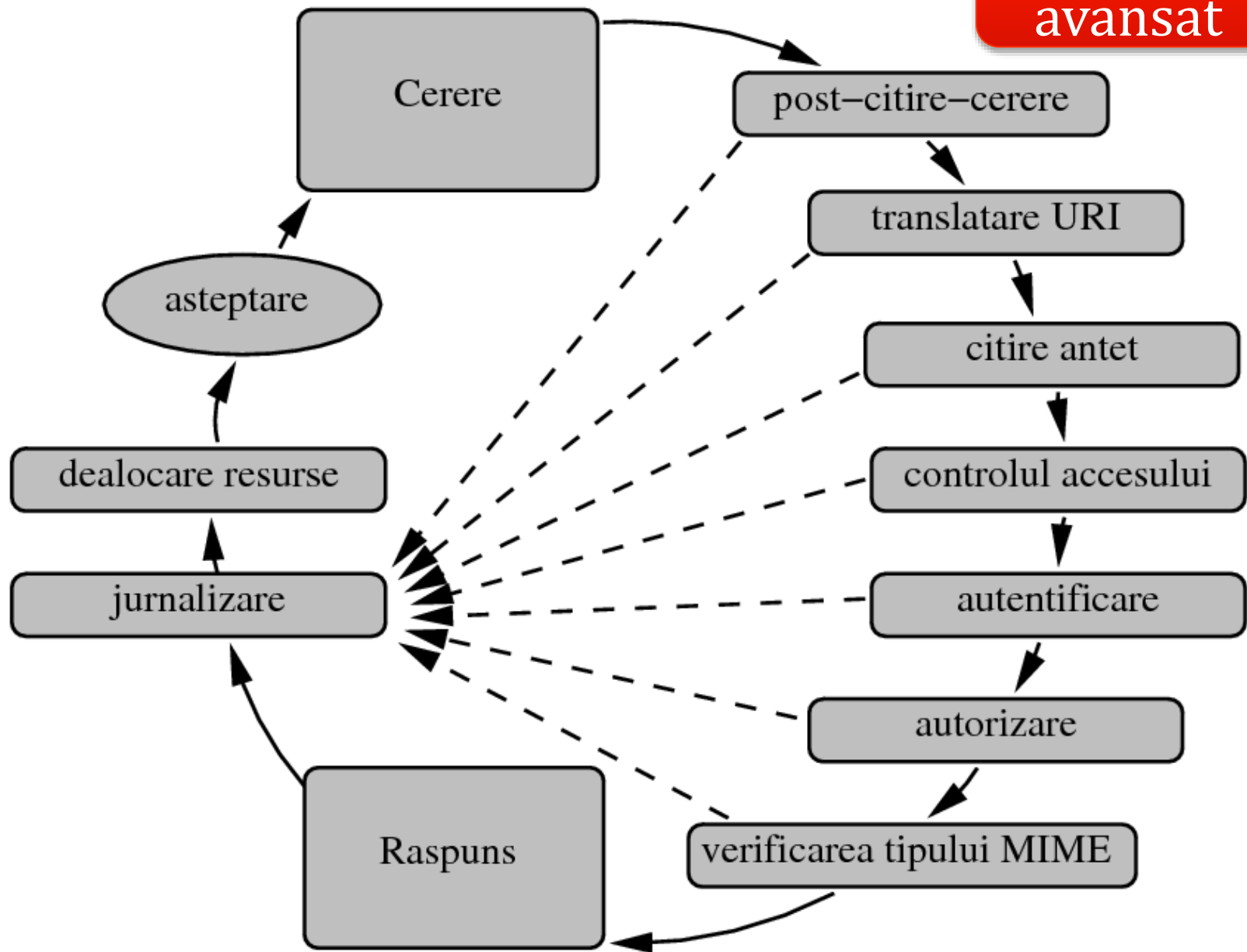
la nivel de utilizator (per director/URI), se poate configura  
via **.htaccess** – vezi și <https://github.com/phanan/htaccess>

# HTTP: server Web

## Studiu de caz: configurarea serverului Apache

posibilitatea de a constitui gazde virtuale (*virtual hosting*)  
același server poate găzdui mai multe situri,  
având diferite nume simbolice

avansat



serverul Apache: bucla de servire a cererilor

# HTTP: server Web

Uzual, arhitectura serverului Web e modularizată

nucleu (*core*)

+

module implementând funcționalități specifice

# HTTP: server Web

Uzual, arhitectura serverului Web e modularizată

nucleu (*core*)

+

module implementând funcționalități specifice

oferă o interfață de programare (API) a modulelor  
în limbajul C

# HTTP: server Web

Uzual, arhitectura serverului Web e modularizată

nucleu (*core*)

+

module implementând funcționalități specifice

exemple pentru Apache: **mod\_auth\_basic**, **mod\_cache**,  
**mod\_deflate**, **mod\_include**, **mod\_proxy**, **mod\_session**, **mod\_ssl**

# HTTP: server Web

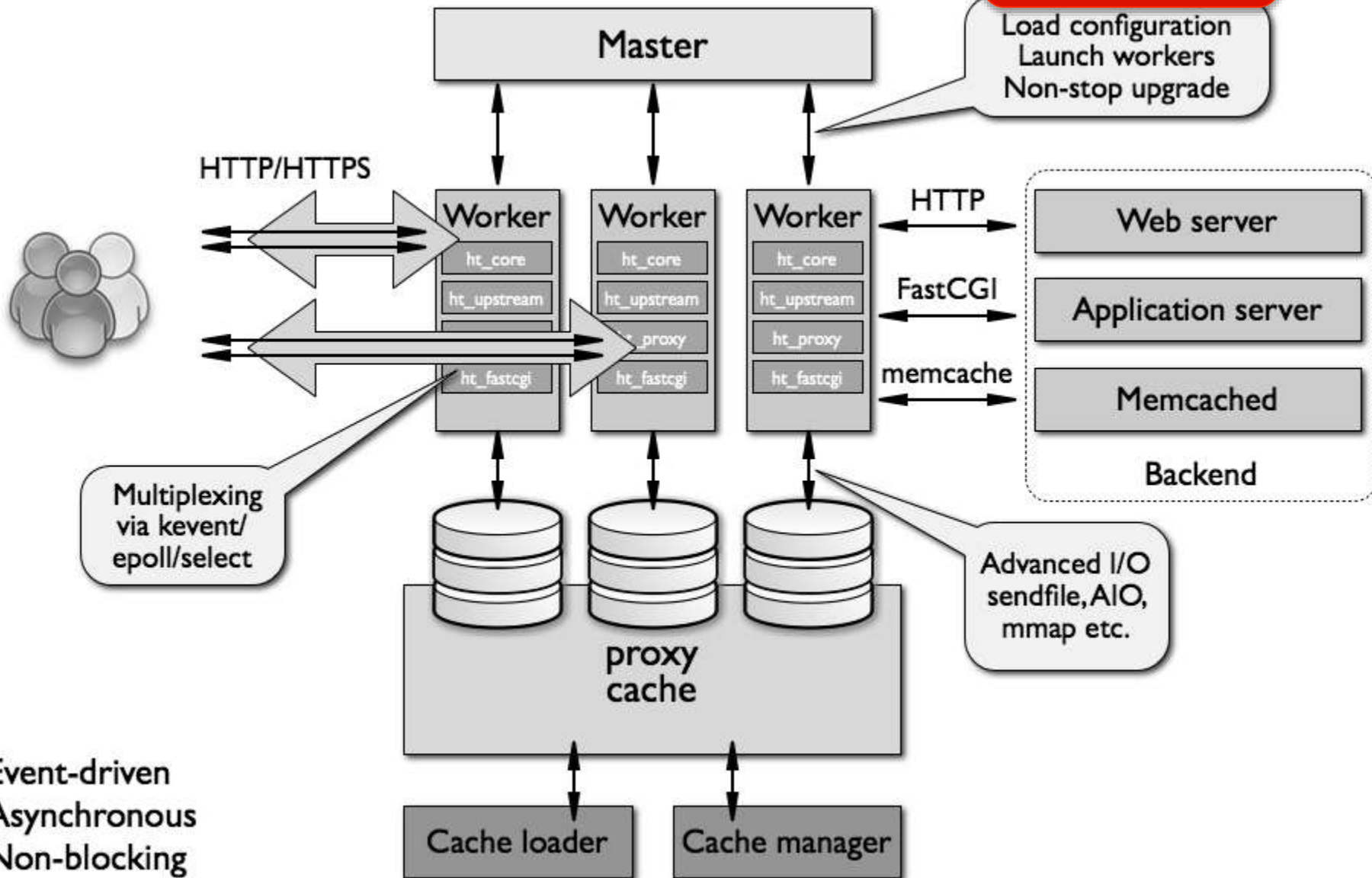
Alternativ, pot fi adoptate strategii *single threaded* asincrone (non-blocante)

exemple de referință:

**nginx**

**Node.js**

avansat



arhitectura serverului Web **nginx**

<http://www.aosabook.org/en/nginx.html>



Cum dezvoltăm aplicații Web  
pe partea de server?

# necesitate

Generarea dinamică la nivel de server  
de reprezentări ale unor resurse  
solicitate de clienții Web

# necesitate

Generarea **dinamică** la nivel de **server**  
de **reprezentări** ale unor resurse  
solicitate de clienții Web

**soluții**

**CGI – *Common Gateway Interface***

**Servere de aplicații Web**

**Cadre de lucru (*framework-uri*) specializate**

# soluții: cgi

Interfață de programare, independentă de limbaj,  
facilitând interacțiunea dintre clienți  
și programe invocate la nivel de server

*standard de facto*

RFC 3875

<http://www.w3.org/CGI/>

# cgi: caracterizare

Un program (*script*) CGI se invocă pe server  
**explicit**

*i.e.*, preluarea informațiilor dintr-un formular Web  
la apăsarea butonului de tip *submit*

# cgi: caracterizare

Un program (*script*) CGI se invocă pe server  
implicit

de exemplu, la fiecare vizită se generează  
o nouă reclamă (*banner* publicitar)

# cgi: caracterizare

*Script*-urile CGI pot fi concepute  
în orice limbaj disponibil pe server

limbaje interpretate

bash, Perl – *e.g.*, modulul Perl::CGI –, Python, Ruby,...

limbaje compilate

C, C++ etc.



# cgi: programare

Orice program CGI va scrie datele  
– reprezentarea resursei Web –  
la ieșirea standard (*stdout*)

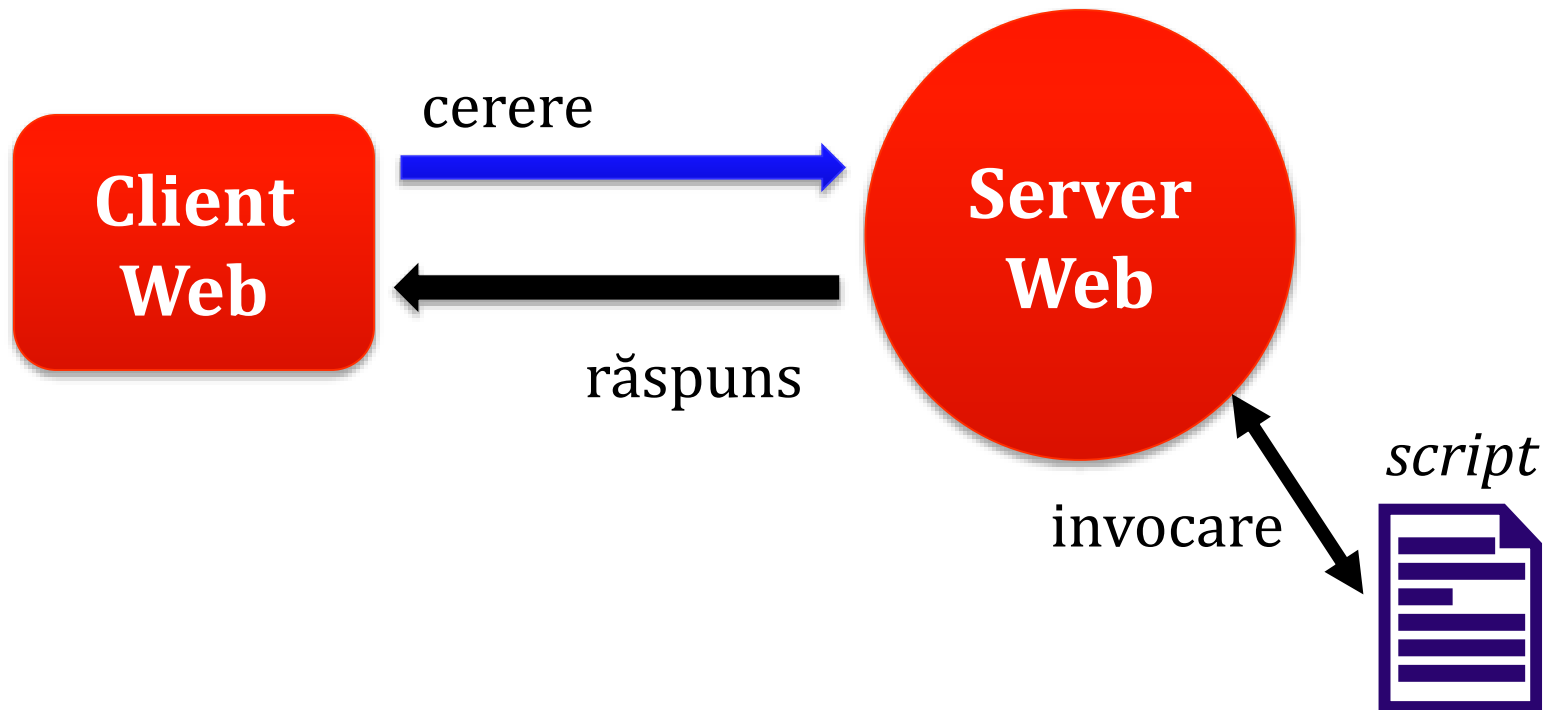
# cgi: programare

Pentru a desemna tipul reprezentării generate, se folosesc anteturi HTTP, recurgându-se la MIME

exemplu: **Content-type:** text/html

# cgi: programare

Interacțiunea dintre clientul și serverul Web



# cgi: variabile

Un *script* CGI are acces la variabile de mediu  
specifice cererilor transmise spre programul CGI:

**REQUEST\_METHOD** – metoda HTTP (GET, POST,...)

**QUERY\_STRING** – șir de interogare: date trimise de client

**REMOTE\_HOST, REMOTE\_ADDR** – adresa clientului

**CONTENT\_TYPE** – tipul conținutului conform MIME

**CONTENT\_LENGTH** – lungimea (în octeți) a conținutului

# cgi: variabile

Variabile suplimentare  
generate, uzual, de serverul Web:

- HTTP\_ACCEPT** – tipurile MIME acceptate de *browser*
- HTTP\_COOKIE** – date despre *cookie*-uri
- HTTP\_HOST** – informații despre gazdă (client)
- HTTP\_USER\_AGENT** – informații privind navigatorul

...și altele

← → ↻

profs.info.uaic.ro/~busaco/cgi/bash/variabile.cgi?date\_de\_intrare

☆ ⌵ ↻

8 cgi

HTTP\_ACCEPT='text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8'

HTTP\_ACCEPT\_ENCODING='gzip, deflate'

HTTP\_ACCEPT\_LANGUAGE='en-gb,en;q=0.5'

HTTP\_CONNECTION=keep-alive

HTTP\_DNT=1

HTTP\_HOST=profs.info.uaic.ro

HTTP\_USER\_AGENT='Mozilla/5.0 (Windows NT 6.3

IFS=\$' \t\n'

MACHTYPE=i486-pc-linux-gnu

OPTERR=1

OPTIND=1

OSTYPE=linux-gnu

PATH=/usr/local/bin:/usr/bin:/bin

PIPESTATUS=( [0]="0" )

PPID=1659

PS4='+ '

PWD=/thor/profs/busaco/html/cgi/bash

QUERY\_STRING=date de intrare

REMOTE\_ADDR=109.100.172.29

REMOTE\_PORT=50088

REQUEST\_METHOD=GET

REQUEST\_URI='/~busaco/cgi/bash/variabile.cgi?date\_de\_intrare'

SCRIPT\_FILENAME=/thor/profs/busaco/html/cgi/bash/variabile.cgi

SCRIPT\_NAME=/~busaco/cgi/bash/variabile.cgi

SERVER\_ADDR=85.122.23.1

SERVER\_ADMIN=admins@info.uaic.ro

SERVER\_NAME=profs.info.uaic.ro

SERVER\_PORT=80

SERVER\_PROTOCOL=HTTP/1.1

SERVER\_SIGNATURE=

SERVER\_SOFTWARE=Apache

SHELL=/bin/bash

#!/bin/bash

# Stabilim tipul conținutului

echo "Content-type: text/plain";

echo

# Executăm comanda 'set' din Linux

# pentru a afișa variabilele de mediu

set

rezultatul obținut de clientul Web în

urma invocării prin **GET** a *script*-ului

variabile.cgi la nivel de server

(având drepturi de citire și execuție)

```
/* hello.c
   (compilare cu gcc hello.c -o hello.cgi) */
#include <stdio.h>

int main() {
    int mesaje;

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

    for (mesaje = 0; mesaje < 10; mesaje++) {
        printf ("<p>Hello, world!</p>");
    }

    return 0;
}
```

programe CGI scrise în C, bash,  
Python generând același  
conținut marcat în HTML

```
#!/bin/bash
# hello.sh.cgi
```

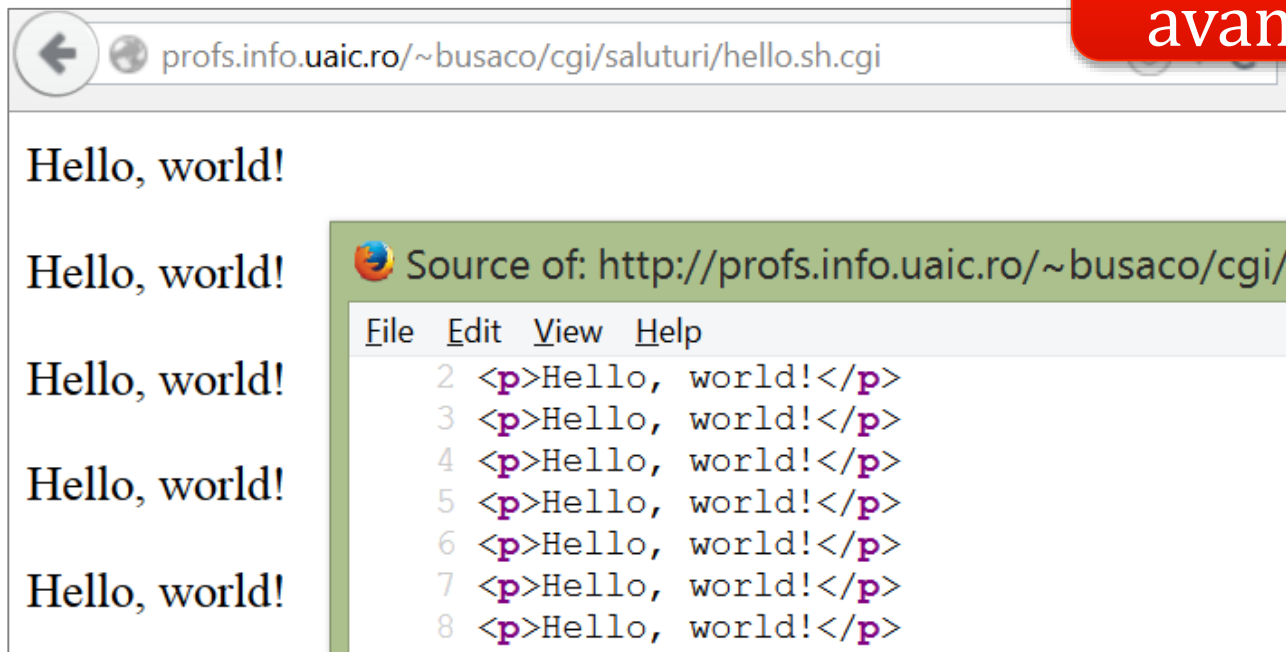
```
echo "Content-type: text/html"
echo
```

```
MESAJE=0
while [ $MESAJE -lt 10 ]
do
    echo "<p>Hello, world!</p>"
    let MESAJE=MESAJE+1
done
```

```
#!/usr/bin/python
# hello.py.cgi
```

```
print "Content-type: text/html\n"
```

```
for mesaje in range (0, 10):
    print "<p>Hello, world!</p>"
```



experimentând și alte tipuri MIME,  
*browser-ul* redă următoarele:

```
<p>Hello, world!</p>
<p>Hello, world!</p>
<p>Hello, world!</p>
<p>Hello, world!</p>
<p>Hello, world!</p>
```

Content-type: **text/plain**

**XML Parsing Error: junk after document element**  
 Location: `http://profs.info.uaic.ro/~busaco/cgi/hello`  
 Line Number 2, Column 1:

```
<p>Hello, world!</p>
^
```

Content-type: **text/xml**

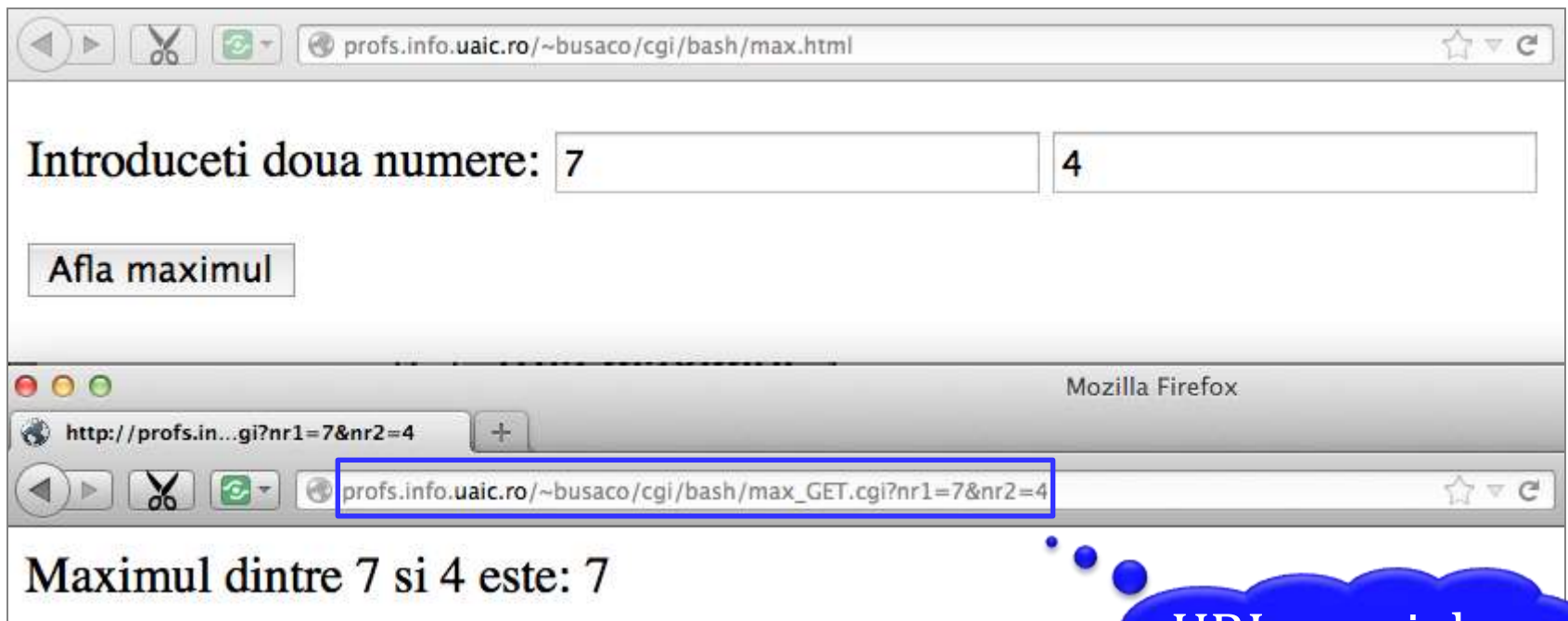


# cgi: invocare

```
<form action="http://www.infoiasi.ro/cgi-bin/max.cgi"
      method="GET">
  <p>Introduceți două numere:
  <input type="text" name="nr1" />
  <input type="text" name="nr2" />
  </p>
  <p><input type="submit" value="Află maximul" /></p>
</form>
```

invocare dintr-un formular interactiv – **GET** sau **POST**

# cgi: invocare



URL special  
cazul GET

# cgi: invocare

Pentru fiecare câmp al formularului, se generează o pereche **nume\_câmp=valoare** delimitată de **&** ce va fi adăugată URL-ului unde e stocat programul

<http://www.infoiasi.ro/cgi-bin/max.cgi?nr1=7&nr2=4>

# cgi: invocare

Exemple concrete:

<http://usabilitygeek.com/?s=design+web>

<https://www.youtube.com/watch?v=hEzmy93zr0Y#t=540>

<https://twitter.com/search?q=web%20development&src=typd>

<https://developer.mozilla.org/search?q=ajax&topic=firefox-os>

acest URL este codificat – *URL encoding*



revezi  
primul curs

# cgi: invocare

Serverul va invoca *script*-ul CGI pasându-i datele  
la intrarea standard  
sau  
via variabile de mediu

# cgi: invocare

Procesarea datelor prin metoda **GET**

date disponibile în variabila de mediu **QUERY\_STRING**

# cgi: invocare

## Procesarea datelor prin metoda **POST**

datele vor fi preluate de la *stdin*, lungimea în octeți a acestora fiind specificată de variabila **CONTENT\_LENGTH**

# cgi: invocare

Procesarea datelor prin metoda **GET** și/sau **POST**

folosind servere de aplicații ori *framework*-uri,  
acestea vor fi încapsulate în structuri de date specifice

ASP.NET (C#) – clasa **HttpRequest**

PHP – tablouri asociative **\$\_GET[]** **\$\_POST[]** **\$\_REQUEST[]**

Play! (Java, Scala) – **play.api.mvc.Request**

Node.js (JavaScript) – **http.ClientRequest**



# GET vs. POST

Metoda **GET** se folosește pentru generarea de reprezentări ale resurselor cerute

*e.g.*, documente HTML, imagini JPEG, fluxuri de știri Atom/RSS, arhive în format ZIP etc.

starea serverului nu trebuie să se modifice

# GET vs. POST

Metoda **GET** se folosește pentru generarea de reprezentări ale resurselor cerute

accesând datele prin GET, utilizatorul poate stabili un *bookmark* pentru acces ulterior la o resursă Web (folosind URL-ul reprezentării resursei generate)

e.g., <https://duckduckgo.com/?q=web+programming&ia=videos>

# GET vs. POST

Metoda **POST** se utilizează atunci când datele transmise serverului au dimensiuni mari (*e.g.*, conținut de fișiere ce a fost transferat prin *upload*) sau sunt „delicate” – exemplu tipic: parole

# GET vs. POST

Metoda **POST** se utilizează atunci când datele transmise serverului au dimensiuni mari (*e.g.*, conținut de fișiere ce a fost transferat prin *upload*) sau sunt „delicate” – exemplu tipic: parole

de asemenea, când invocarea programului poate conduce la modificări ale stării pe server: adăugarea unei înregistrări, alterarea unui fișier,...

## cgi: suport

Serverul Web trebuie să ofere suport pentru invocarea de *script*-uri CGI

de exemplu, la nivelul serverului Apache se utilizează modulul **mod\_cgi**

# cgi: ssi

*Script*-urile CGI pot fi invocate direct dintr-un document HTML via *SSI* (*Server Side Includes*)

<http://www.ssi-developer.net/ssi/>

Apache: <http://httpd.apache.org/docs/trunk/howto/ssi.html>

Nginx: [http://nginx.org/en/docs/http/nginx\\_http\\_ssi\\_module.html](http://nginx.org/en/docs/http/nginx_http_ssi_module.html)

# cgi: fastcgi

## FastCGI

alternativă la CGI focalizată asupra performanței

[www.fastcgi.com](http://www.fastcgi.com)

exemplificări:

suport pentru diverse limbaje (D, PHP, Python, Ruby,...)  
și servere (Apache, IIS, Lighttpd, Nginx etc.)

[www.fastcgi.com/drupal/node/5](http://www.fastcgi.com/drupal/node/5)

Există o manieră prin care se pot stoca  
– temporar –, la nivel de client (*browser*),  
date trimise de aplicația Web de pe server?



# cookie-uri

Mecanism standard ce permite ca un server Web să plaseze date pe calculatorul-client (la utilizator), prin intermediul *browser*-ului, pentru ca, ulterior, navigatorul să returneze acele date aceluiasi server

# *cookie-uri*

Mijloc persistent de stocare a datelor pe mașina clientului Web cu scopul de a fi apoi accesate de un program rulând pe server

# cookie-uri: utilizări

Memorarea preferințelor fiecărui utilizator

exemple tipice:

opțiuni vizând interacțiunea – temă vizuală  
(*e.g.*, cromatică), preferințe lingvistice etc.

localizare geografică, interese privind cumpărăturile

...

# *cookie-uri: utilizări*

Completarea automată a formularelor

folosirea valorilor introduse anterior de utilizator  
în anumite câmpuri

# *cookie-uri: utilizări*

Monitorizarea accesului la o resursă Web

aspect de interes:

***Web analytics***

colectarea de informații despre clienți  
(platformă hardware, *browser*, rezoluție etc.)

# *cookie-uri: utilizări*

Monitorizarea accesului la o resursă Web

aspect de interes:

*user tracking*

monitorizarea comportamentului utilizatorului

► inițiativa *Do Not Track* – <http://donottrack.us/>

# *cookie-uri: utilizări*

Stocarea informațiilor de autentificare

*e.g.*, reținerea datelor privitoare la contul utilizatorului  
în contextul comerțului electronic

# *cookie-uri: utilizări*

Starea tranzacțiilor în cadrul unei aplicații Web

*e.g.*, starea coșului de cumpărături  
în cadrul unui magazin virtual (*e-shop*)



*cookie-uri: utilizări*

Managementul sesiunilor Web

# *cookie-uri: tipuri*

## *Cookie-uri persistente*

nu vor fi distruse la închiderea navigatorului Web, ci vor fi memorate într-un fișier, perioada lor de viață fiind stabilită de creatorul *cookie*-urilor

# *cookie-uri: tipuri*

*Cookie-uri nepersistente*

dispar la închiderea *browser*-ului

# cookie-uri

Un *cookie* poate fi considerat ca fiind o variabilă a cărei valoare este vehiculată via HTTP între server (aplicația) Web și client (*browser*)

# cookie-uri

Un *cookie* poate fi considerat ca fiind o variabilă a cărei valoare este vehiculată via HTTP între server (aplicația) Web și client (*browser*)

constă dintr-o pereche *nume=valoare*

valoarea este un șir de caractere *URL-encoded*

# *cookie-uri*

Datele referitoare la un *cookie* vor fi recepționate de navigator care menține o listă de *cookie-uri* aparținând serverului care le-a trimis

# cookie-uri

Un *cookie* este trimis unui client  
folosind câmpul **Set-Cookie**  
dintr-un antet al unui mesaj de răspuns HTTP

# cookie-uri

**Set-Cookie:** *nume=valoare*; expires=*data*; path=*cale*;  
domain=*domeniu*; secure



# cookie-uri

**Set-Cookie:** *nume=valoare*; expires=*data*; path=*cale*;  
domain=*domeniu*; secure

expires – indică data și timpul  
când *cookie*-ul va expira, iar clientul Web îl va distruge

# cookie-uri

**Set-Cookie:** *nume=valoare*; expires=*data*; path=*cale*;  
domain=*domeniu*; secure

**domain** – semnifică numele simbolic al serverului Web  
care a generat *cookie*-ul

# cookie-uri

```
Set-Cookie: nume=valoare; expires=data; path=cale;  
            domain=domeniu; secure
```

*path* – specifică un subset de URL-uri  
din domeniul corespunzător unui *cookie*

diferențiază aplicații multiple existente pe același server

# cookie-uri

**Set-Cookie:** *nume=valoare*; expires=*data*; path=*cale*;  
domain=*domeniu*; secure

**secure** – indică faptul că acest *cookie* va fi transmis doar în cazul în care canalul de comunicație este „sigur”  
(via HTTPS)

Search:



The following cookies are stored on your computer:

Site	Cookie Name
■ windowsazure.com	
□ windowsazure.com	geoc
▶ windowsphone.com	
▶ wired.com	
▶ wireframe.cc	
■ wordpress.com	
□ wordpress.com	wordpress
□ wordpress.com	wordpress
□ wordpress.com	wordpress_logged_in
□ wordpress.com	wordpress_sec
□ wordpress.com	wordpress_sec
□ wordpress.com	anonymous_stats

Name: wordpress

Content: busaco%7C1487146849%7C6c33c84ffb5bfe7923

Domain: .wordpress.com

Path: /wp-content/plugins

Send For: Any type of connection

Expires: Wednesday, 15 February, 2017 10:20:50

Remove Cookie

Remove All Cookies

Close

# cookie-uri

Un *cookie* este transmis înapoi de la client spre serverul Web doar dacă îndeplinește toate condițiile de validitate

se potrivesc domeniul,  
calea (virtuală) de directoare, timpul de expirare  
și securitatea canalului de comunicație

# cookie-uri

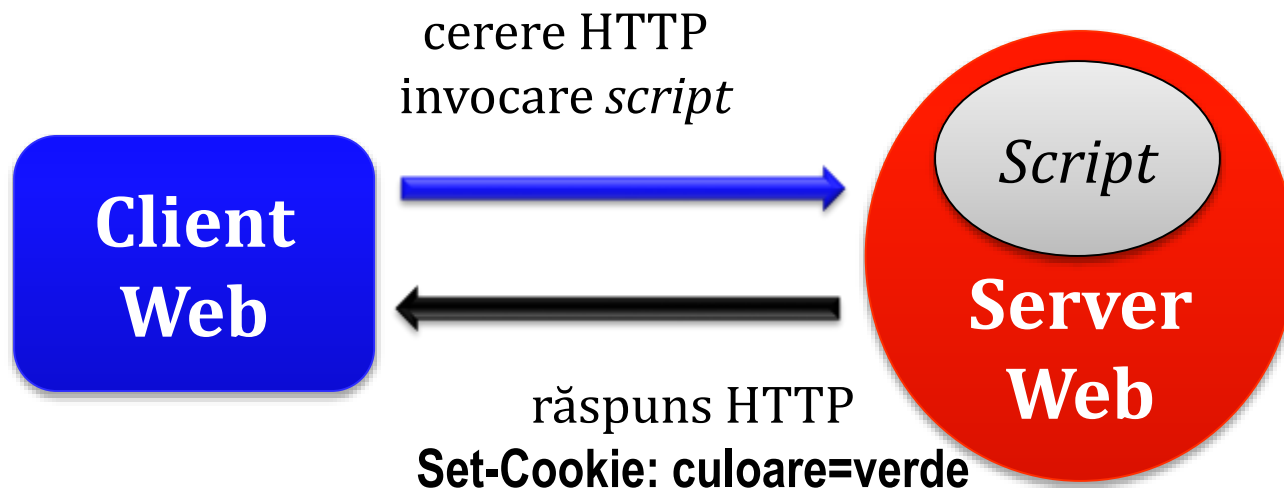
Serverul va primi de la client,  
în antetul unui mesaj HTTP, o linie de forma:

**Cookie:** nume1=**valoare1**; nume2=**valoare2**...

lista *cookie*-urilor ce respectă condițiile de validitate

# cookie-uri

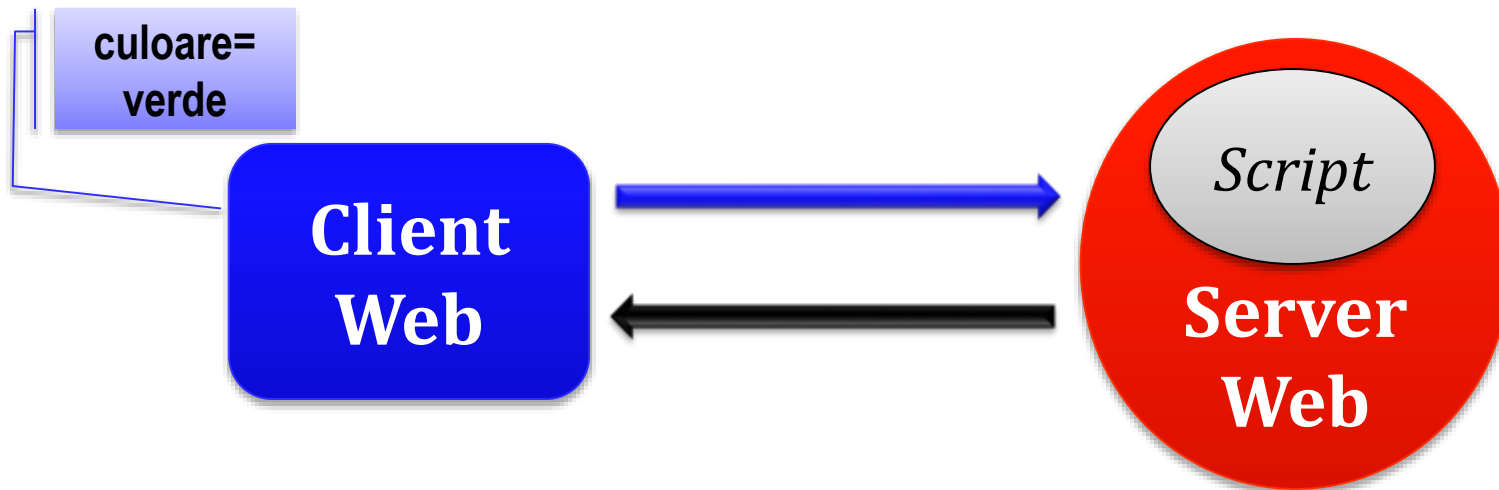
Invocarea *script*-ului conduce la returnarea unei reprezentări + setarea de *cookie*-uri





# cookie-uri

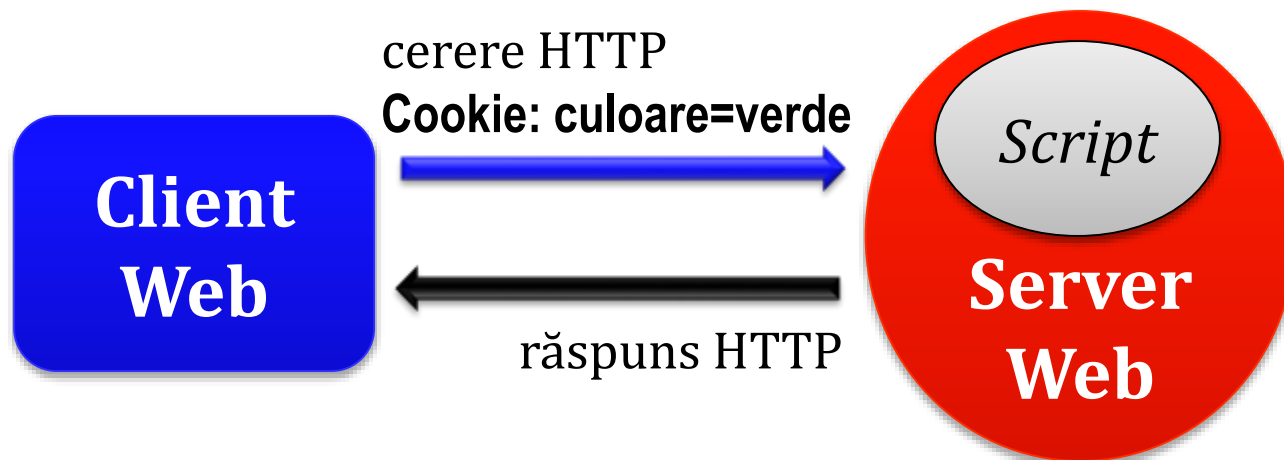
*Cookie*-urile – persistente sau nu – sunt memorate la nivel de *browser*



*cookie*-uri persistente stocate în fișiere sau baze de date (SQLite)

# cookie-uri

Următorul acces la *script* se face cu transmiterea  
*cookie*-urilor spre server  
conform condițiilor de validitate



## cookie-uri: creare

Exemplu – pentru CGI, folosind bash:

```
#!/bin/bash  
echo "Set-Cookie: culoare=verde; path=/  
      expires=Mon, 25-May-2015 00:33:00 GMT"  
...
```

# cookie-uri: creare

Exemplu în cazul PHP – funcția **setcookie ()**

```
<?php
    setcookie ("alta_culoare", "albastra");
    echo "Un cookie de culoarea " . $_COOKIE["alta_culoare"];
?>
```

# *cookie-uri: expirare*

Se anulează valoarea și timpul,  
eventual anulându-se și celelalte atribute ale *cookie*-ului

exemplu – pentru PHP:

```
<?php  
    setcookie ($nume_cookie, "", 0, "/", "", 0);  
?>
```

## *cookie-uri*: consultare

*Cookie*-urile se regăsesc în câmpul din antetul unui mesaj vehiculat via protocolul HTTP

**HTTP\_COOKIE**

# cookie-uri: consultare

## Cazul PHP

*cookie*-ul e specificat (accesat) ca variabilă:

**`$_COOKIE`** `['nume_cookie']`



tablou asociativ

# cookie-uri: consultare

## Exemplu pentru CGI – *script* Perl

```
@cookieuri = split (/;/, $ENV{'HTTP_COOKIE'});  
foreach $pereche (@cookieuri) {  
    ($nume, $val) = split (/=/, $pereche);  
    $cookie{$nume} = $val; # tablou asociativ  
}  
$fundal = $cookie{'alta_culoare'};
```



# cookie-uri: manipulare

Pentru ASP.NET: proprietatea **Cookies**  
a colecției **HttpCookieCollection**  
(vezi **HttpRequest** și **HttpResponse** din **System.Web**)

```
HttpCookie vizita = new HttpCookie ("ultima_vizita");  
DateTime timp_curent = DateTime.Now;  
vizita.Value = timp_curent.ToString ();  
vizita.Expires = timp_curent.AddHours (24); // expiră peste 1 zi  
Response.Cookies.Add (vizita);
```

# cookie-uri

Alte informații de interes sunt disponibile în  
RFC 6265

*HTTP State Management Mechanism*

<http://tools.ietf.org/html/rfc6265>

Cum identificăm cereri succesive  
formulate de aceeași instanță a clientului?



# preliminarii

HTTP este un protocol *stateless*,  
neputând oferi informații dacă anumite cereri  
succesive provin de la același client  
(eventual, de la aceeași instanță a navigatorului)

# preliminarii

Apare necesitatea de a prezerva anumite date  
de-a lungul mai multor accesări înrudite

# preliminarii

Apare necesitatea de a prezerva anumite date de-a lungul mai multor accesări înrudite

exemple:

starea coşului de cumpărături,  
formulare Web completate în mai mulţi paşi,  
paginarea conţinutului,  
starea autentificării utilizatorului  
etc.

# sesiuni

Orice vizitator al sitului va avea asociat un identificator unic – **session ID (SID)**

stocat într-un *cookie*

(*e.g.*, ASP.NET\_SessionId, PHPSESSID, session-id, \_wp\_session)

ori

propagat via URL

# sesiuni

Orice vizitator al sitului va avea asociat un identificator unic – **session ID (SID)**

astfel, se pot identifica vizite (cereri) consecutive realizate de același utilizator



# sesiuni

Unei sesiuni i se pot asocia diverse variabile

ale căror valori vor fi menținute (păstrate)  
între accesări consecutive – *e.g.*, înrudite –  
din partea aceleiași instanțe  
a clientului (*browser*-ului) Web

# sesiuni

O sesiune se poate înregistra (iniția) implicit sau explicit, în funcție de serverul de aplicații ori de configurația prestabilită

# sesiuni

O sesiune se poate înregistra (iniția) implicit sau explicit, în funcție de serverul de aplicații ori de configurația prestabilită

uzual, informațiile despre sesiuni sunt stocate persistent la nivel de server via servere de baze de date

- *e.g.*, DynamoDB, Memcached, PostgreSQL, Redis,... – ori în cadrul sistemului de fișiere

GET / HTTP/1.1

Host: [plus.google.com](https://plus.google.com)

User-Agent: Mozilla/5.0 ... Firefox/35.0

Accept: text/html,application/xhtml+xml;q=0.9,\*/\*;q=0.8

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip, deflate

Cookie: **PREF**=ID=d7ab9cd8e5e1f17d:U=213249fe7b4fe3ea:....;

**SID**=DQABANMAAAC5eFmWHrrpw203...iK4g;

**HSID**=AlJExygxk\_sNdOlAX;

**SSID**=ATkVPS5BM3xEjGwXZ;

**APISID**=cXJw6EX3K\_WAUyDd/A9wP-XcSkrjWocGEe;

**SAPISID**=B13Zj7NiRE23qoe4/AlVGLP6D5ckMamKjz;

**OTZ**=1136908\_48\_48\_123900\_44\_436380;

**SS**=DQABANMAAADjUh...MKA;

**ULS**=EgYKBBICZW4YuLGa-gQ

*cookie*-uri (stocând inclusiv informații despre sesiunea curentă) într-o cerere GET

HTTP/1.1 200 OK

**Set-Cookie:** **SID=DQABANMAAAC5eFmWHrrpw203...iK4g;**  
**Domain=.google.com;**  
**Path=/;**  
**Expires=Mon, 02-Mar-2025 13:24:40 GMT**

**Content-Type:** text/html; charset=utf-8

**Content-Encoding:** gzip

**Date:** Fri, 20 Feb 2015 13:24:40 GMT

**Server:** GSE

<!DOCTYPE html>

...

răspunsul HTTP incluzând  
setarea *cookie*-ului privitor la sesiunea Web

## sesiuni: programare


În cazul CGI, managementul sesiunilor cade în responsabilitatea programatorului

nu există o manieră standardizată de gestionare a sesiunilor Web

# sesiuni: programare

PHP: funcțiile `session_start()`, `session_register()`, `session_id()`, `session_unset()`, `session_destroy()`

```
<?php
session_start (); // inițiem o sesiune
if (!isset ($_SESSION['accesari'])) {
    $_SESSION['accesari'] = 0; } else {
    $_SESSION['accesari']++; }
?>
```



variabila  
accesari atașată  
sesiunii

detalii la <http://php.net/manual/en/book.session.php>

# sesiuni: programare

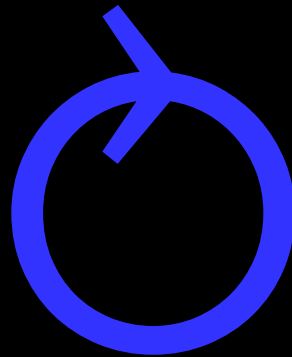
Folosind un server de aplicații ori un *framework*, managementul *cookie*-urilor și sesiunilor e simplificat

diverse exemplificări:

clasa `HttpSession` (ASP.NET), interfața `HttpSession` (*servlet*-uri Java),  
`HTTP::Session` (Perl), `session` (oferit de Flask – *framework* Python),  
`HttpFoundation` (componentă Symfony – *framework* PHP),  
clasa `SessionComponent` (CakePHP), `session` (tablou Ruby on Rails),  
`play.mvc.Http.Cookie` (Play! pentru Java/Scala),  
`cookie-parser` și `express-session` (module Node.js pentru Express)



# rezumat



protocolul HTTP + arhitectura serverelor Web  
*cookie-uri* și sesiuni Web



HTML, CSS, SVG, WebGL etc.

***View***

la nivel client(i) Web



SQL, NoSQL, XML (XQuery), RDF,...

***Model***

stocare persistentă



servere de aplicații, *framework*-uri

***Controller***

aplicație Web la nivel de server

episodul viitor: **programare Web**

servere de aplicații Web, arhitectura aplicațiilor Web