Curs 7
HTTP

Gestiunea serviciilor de rețea (GSR)
17 noiembrie 2016

Departamentul de Calculatoare, Comunitatea RLUG

Cuprins

Un pic de istorie

- Dezvoltat inițial de către Sir Tim Berners Lee la `CERN`
- `HTTP/0.9` a fost prima versiune a protocolului, publicată în 1991
- `HTTP/1.0` reprezintă o îmbunătățire adusă protocolului în 1996 prin `RFC1945`
- `HTTP/1.1` a fost standardizat în anul 1999 prin `RFC2616`
- În 2014 au fost publicate ultimele revizii ale protocolului HTTP/1.1:
  - `RFC7230`: Message Syntax and Routing
  - `RFC7231`: Semantics and Content
  - `RFC7232`: Conditional Requests
  - `RFC7233`: Range Requests
  - `RFC7234`: Caching
  - `RFC7235`: Authentication

Concepte

- HTML: Hyper Text Markup Language
- HTTP: Hyper Text Transfer Protocol
- URL: Uniform Resource Locator

Cum funcționează HTTP

- Protocol `stateless` pe bază de comenzi simple între `client` și `server`
  - Clientul efectuează o cerere către server pentru o resursă
  - Serverul returnează un răspuns pentru resursa respectivă
  - + informații trimise în antete, atât de `client`, cât și de către `server`
- Fiecare pereche cerere/răspuns folosește o conexiune separată, serverul închizând socket-ul după ce trimite răspunsul.
- Comenzi:
  - `GET`
  - `POST`
  - `HEAD`
  - Ex.: `GET / HTTP/1.0 \n\n`
- Mesajele protocolului (headere) sunt plaintext, nu neaparat și contentul

Comanda GET

- Comanda `GET` este folosită pentru a cere o resursă de la server

```
GET /cale/către/resursă HTTP/1.1
```

- Ca și exceptie, atunci când clientul accesează prima oara un server

```
GET / HTTP/1.1
```

Comanda POST

- Comanda `POST` este folosită pentru a trimite date către server

```
POST /cale/către/resursă HTTP/1.1
```

Comanda HEAD

- Comanda `HEAD` se folosește identic cu comanda `GET`, însă răspunsul va conține doar antetele trimise de server, fără a primi însă și conținutul efectiv al resursei accesate
- O folosim atunci când ne interesează doar informațiile din antet (data ultimei modificări, mărimea resursei etc.)

Antete client

- Informații trimise de către `client` spre `server`
  - `User-Agent`

    Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/537.36 (KHTML, like Gecko)
  - `Accept`

    text/html,application/xhtml+xml,application/xml
  - `Accept-Encoding`

    gzip, deflate
  - `Accept-Language`

    en-US,en;q=0.8,ro;q=0.6
  - Host
  - If-Modified-Since
  - Connection
  - Cookie
  - X-Forwarded-For

Antete server

- Informații trimise de server către client
  - `Content-Encoding`

    ```
    Content-Encoding: gzip
    ```
  - `Content-Type`

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

    ```
    Content-Length: 30286
    ```
  - `Cache-Control`

    ```
    Cache-Control: max-age=60
    ```
  - Content-Location
  - Set-Cookie
  - Server
  - Expires

Cookies (1)

- `Cookies` reprezintă informații trimise de către server clientului, pe care clientul trebuie să le trimită nemodificate înapoi la server atunci când face o conexiune ulterioară la acesta
- Formatul unui `Cookie` este de forma `Cheie = Valoare` (text ASCII)
- La ce folosesc?
  - Salvarea preferințelor utilizatorilor
  - Transmiterea informațiilor despre sesiuni
  - Urmărirea utilizatorilor
  - Salvarea informațiilor de autentificare
- Se setează per domeniu

Cookies (2)

- Primire `Cookie` de la server

```
Set-Cookie:  CNNtosAgreed=true
Set-Cookie:  countryCode=BH
Set-Cookie:  ug=5718c85f0787e40a3c743c2d5803087b
```

- Trimitere `Cookie` către server

```
Cookie:  CNNtosAgreed=true; countryCode=BH;
ug=5718c85f0787e40a3c743c2d5803087b
```

Coduri de stare

- Similar cu `SMTP`, protocolul `HTTP` folosește coduri de stare pentru a interacționa cu clienții
- `1xx`: Informational
- `2XX`: Succes
- `3XX`: Redirectări
- `4XX`: Eroare de client
- `5XX`: Eroare de server

2XX - Succes

- 200: OK
- 203: Non-Authoritative Information
- 204: No Content
- 206: Partial Content

3XX - Redirectări

- 301: Moved Permanently
- 302: Found
- 303: See Other
- 304: Not Modified
- 307: Temporary Redirect
- 308: Permanent Redirect

4XX - Erori de client

- 400: Bad Request
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found
- 416: Range Not Satisfiable
- 418: I'm a teapot
- 431: Request Header Fields Too Large

5XX - Erori de server

- 500: Internal Server Error
- 502: Bad Gateway
- 503: Service Unavailable
- 504: Gateway Time-out

- Cea mai simplă metodă de autentificare la un server HTTP

  ... însă și cea mai nesigură

```
S: HTTP/1.1 401 Access Denied
S: WWW-Authenticate:  Basic realm="GSR"

C: GET /cale/resursă/protejată HTTP/1.1
C: Authorization:  Basic
ZXVnZW46Y2VhbWFpdGFyZXBhcm9sYQ==
```

- Se poate face per resursă, în mod arborescent (nested)

Formatul unui URL

```
scheme:[//[user:password@]domain[:port]][/]path[?query][#fragment]
```

- O ,,pagină web" implică mai multe resurse încarcate în cascadă de către browser

Cum știe un server să se ,,prindă” cine e userul și ce-a mai făcut în trecut?

- **IP address**
- **HTTP authentication**
- **query string parameters**
  - Linkuri generate de forma: `http://site.com/page.html?session=b026324c6904b2a9cb4b88d6d61c81d1`
- **Cookies**

Cuprins

Exemplu request

```
GET /images/logo.jpg HTTP/1.1
Host: www.mycompany.com
User-Agent: Example Browser 1.0
Cookie: lang=ro; session=VbXZgH3P46OoDcFJZIsN8024siBFpMrW
If-Modified-Since: Thu, 22 Oct 2015 13:56:31 GMT
Connection: close
```

1. Decide dacă are o configurație anume pentru hostul din Host:
2. Se uită ce fel de reguli are pentru resursa indicată (fișier static)
3. Compune calea pe disc (să zicem "/var/www"+"/images/logo.jpg")
4. Vede dacă există fișierul și data modificării lui
5. Compune raspunsul

Fisier inexistent

```
HTTP/1.1 404 Not Found
Date: Thu, 22 Oct 2015 16:21:42 GMT
Server: Nginx/1.1
Content-Length: 134
Content-Type: text/html; charset=utf-8

<html><head>
<title>File not found</title>
</head><body>
Requested file wasn't found
</body></html>
```

Fisier gasit

```
HTTP/1.1 200 OK
Date: Thu, 22 Oct 2015 16:21:42 GMT
Server: Nginx/1.1
Content-Length: 1245
Content-Type: image/jpg
Last-Modified: Thu, 22 Oct 2015 14:47:28 GMT

...binary data follows...
```

Fisier gasit, nemodificat

```
HTTP/1.1 304 Not Modified
Date: Thu, 22 Oct 2015 16:21:42 GMT
Server: Nginx/1.1
```

Cuprins

Anything over HTTP

- Datorită simplității și versatilității sale, peste HTTP s-au implementat metode de comunicație între aplicații
  - ReST
  - SOAP
  - XML-RPC
- Aceste metode mai sunt cunoscute în mod generic și sub numele de `Web Services`

Web Server Stack

- In aplicatiile web moderne, un webserver nu mai este folosit doar pentru a citi o resursa de pe disc si a o servi clientului, ci face parte din ceea ce se cheama ,,application stack”
- Un ,,application stack” este format din
  - Client (browser)
  - Reverse proxy / Load balancer
  - Web Server
  - Application Server(s)
  - Database Service(s)
  - Key/Value store

Cuprins

HTTP/2

- Revizie majoră a protocolului HTTP în decembrie 2014
- Bazat pe protocolul SPDY, dezvoltat de Google
- Scopul HTTP/2 este sa reducă timpul de load al paginilor web
  - Modificare compatibilă cu HTTP/1.1
  - Compresia headerelor
  - Server content push
  - Multiplexarea mai multor cereri peste aceeași conexiune TCP
- În octombrie 2016, 10% din tip 10M websites suportă HTML/2

Resurse

- https://en.wikipedia.org/wiki/List_of_HTTP_header_fields
- https://en.wikipedia.org/wiki/Basic_access_authentication
- https://en.wikipedia.org/wiki/HTTP/2