EL PROTOCOL HTTP

ÍNDEX

Introducció

- Segons el document RFC 1945 l'IETF, el HTTP és un protocol de nivell d'aplicació lleuger i ràpid, que permet la comunicació sobre internet de sistemes d'informació distribuïts, col·laboratius i hypermedia. En paraules una mica més senzilles, és el protocol d'internet que s'utilitza per intercanviar dades en forma de gràfics, audio, video, texte, html, resultats d'un query i hyperlinks sobre WWW. De fet, WWW funciona utilitzant el protocol HTTP des dels 90s del segle passat.

Introducció

- Les caracterísitques principals del protocol HTTP són:
  - **És un protocol connectionless:** Generalment, el client estableix una connexió (pràcticament sempre de tipus TCP) i envia un missatge de petició HTTP al servidor i a continuació, espera fins que arribi una resposta. El servidor processa la petició i envia la resposta al client. A continuació, el client tanca la connexió. Una nova petició requereix establir una nova connexió des del principi com si mai s'hagués previament una connexió.
  - **És un protocol sense estats (stateless):** No es desa a cap lloc informació sobre les anteriors connexions, ni de les dades enviades, ni res de qualsevol connexió que s'hagués establer en el passat. El client i servidor no retenen res. Cada cop que es tanca una connexió, el servidor i el client obliden qualsevol connexió i intercanvi de dades que hagin realitzat.

Introducció

- Les caracterísitques principals del protocol HTTP són:
  - **És un protocol independent del tipus de dades:** Això vol dir que pot servir per transportar dades de qualsevol tipus. En tot cas el client i el servidor ja sabran que fer amb les dades. Per HTTP les dades no tenen tipus no són res més que un munt de 0 i 1 sense format.
  - **És un protocol que utilitza el model client/servidor:** La comunicació és realitza entre un programa client (generalment un navegador) que realitza una petició d'accés a un recurs al servidor (per exemple Apache2), el servidor analitza la petició, busca i troba el recurs i envia un missatge de resposta amb el recurs al client.
  - Més informació aquí.

Introducció

- El protocol HTTP ha passat per diverses versions i actualitzacions importants que es publiquen al dipòsit de documents RFC escrits principalment per l'IETF :
  - HTTP/1.0 - Maig del 1996 / RFC 1945
  - HTTP/1.1 - Gener del 1997 / RFC 2068
  - HTTP/1.1 (Actualització) - Juny 1999 - RFC 2616. Aquest document deixa obsolet el RFC 2068
  - HTTP/1.1 (Actualització) - Juny 2014 - RFC 7230 a 7240. Aquests documents deixen obsolet el RFC 2616.

Introducció

- Existeix una versió HTTP/0.9 que no va arribar a ser estàndaritzada dins dels documents RFC.
- Si voleu llegir algunes de les millores que s'han anat introduïnt des de la versió HTTP/0.9 a la versió HTTP/1.1 de manera resumida podeu llegir aquest enllaç sobre l'evolució de HTTP.
- Enllaços a dipòsits de documents RFC:

  http://www.w3.org/Protocols

  http://www.rfc-editor-org/rfc-indexhtm

  https://www.rfc-editor.org/search/rfc_search.php

Introducció

- El mes de Maig de l'any 2015, el document RFC 7540 introdueix el nou protocol HTTP/2 que:

  a) No deixa obsolets el documents RFC 7230 a 7240.

  b) És compatible amb HTTP/1.1. Característiques molt important com els mètodes, codis d'estat, format de les capçaleres, format URL per exemple són iguals en HTTP/1.1 i HTTP/2

  c) Decrementa la latència i millora la velocidad de carrega de les pàgines en els navegadors.

Introducció

d) No cal fer canvis en les aplicacions existents per treballar amb HTTP/2 i les noves aplicacions poden aprofitar les noves caracterísitques per millorar la velocitat de navegació.

e) Minificació, HTTP/2 Server Push, Compressió de les capçaleres, pipelining, multiplexació de multiples peticions i millores en la solució del problema de head-of-blocking són algunes característiques importants de HTTP/2

f) HTTP/2 serà estudiada en profunditat amb en Sergi Grau als mòduls M06, M07 i M14.

Uniform Resource Identifier

L'especificació de la definició d'URI es troba a la RFC 3986. La forma general d'un URI es pot expressar de la forma següent:

esquema: identificador

- En HTTP les URI bàsicament identifiquen un recurs a partir del nom del servidor, el port de comunicacions, la ruta, etc..El seu format està definit a la secció 2.7 del document RFC7230 (http://tools.ietf.org/html/rfc7230#section-2.7)
- En HTTP quan es parla de localització de recursos s'utilitza el tipus específic d'URI anomenat **URL** (Uniform Resource Locator).
- Més informació aquí.

HTTP URL

- La URL amb l'esquema HTTP és un cas específic d'URI i es representa de la següent manera:

http_URL = "http:" "//" host [":" port] [abs_path ["?" query]]

- Per exemple: http://www.w3.org/protocols
  - El host = www.w3.org (podria ser una IP)
  - El port si no s'especifica correspon al 80
  - El camí absolut al recurs = /protocols
  - En aquest cas no hi ha query.

Tipus i format del missatges HTTP

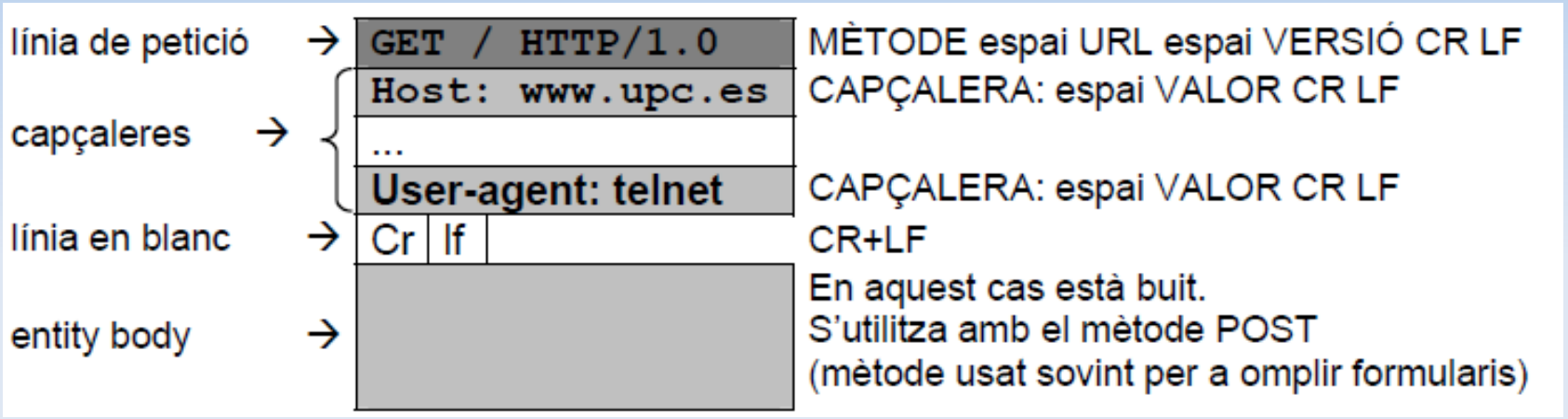
- D'acord amb el document RFC 7230 secció 3 (http://tools.ietf.org/html/rfc7230#section-3), els missatges HTTP poden ser de dos tipus:
  - **Petició** (request) del client al servidor
  - **Resposta** (response) del servidor al client.
- Els missatges consisteixen en:
  - Una **línia inicial** (diferent en el cas de la petició o de la resposta)
  - Camps de **capçalera** (headers fields)
  - El **cos del missatge** (especificat al RFC 822)
- Entre el camps de capçalera i el cos del missatge hi ha una línia en blanc (CRLF)

Tipus de missatges HTTP

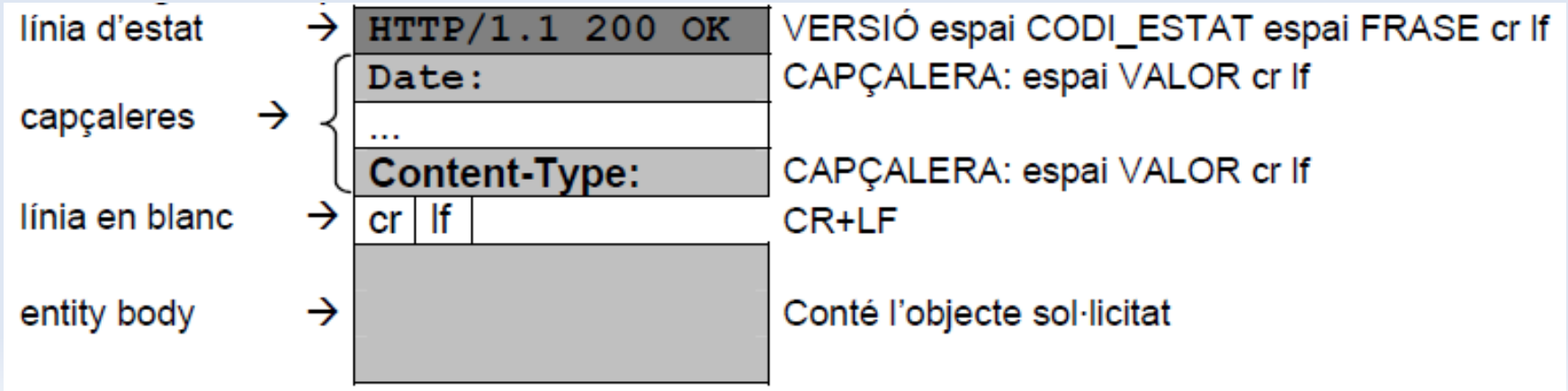
- El cos del missatge, juntament amb els camps de la capçalera que proporcionen informació sobre el seu contingut, formen el que en l'HTTP es denomina una **entitat**.
- Depenent de si el missatge HTTP és una petició o una resposta, la primera línia rep el nom de l**ínia de petició** o **línia d'estatus**, respectivament.

Tipus de missatges HTTP

- **Format missatge de petició:**



- **Format missatge de resposta:**



Missatge petició (request)

- La **línia inicial** dels missatges de petició, anomenada **línia de petició**, té el següent format:

mètode <SP> URI <SP> versió <CRLF>

- El **mètode** especifica el tipus d'operació que solicita el client a fet una petició. El document Doument RFC 7231 secció 4 defineix els diferents mètodes. Els més important són **GET**, **PUT** i **POST**.
- **L'URI** especifica el recurs al qual s'aplica el mètode seleccionat i la **versió** correspon a la versió del protocol.

Missatge petició - Exemple

Per exemple per obtenir el recurs *principal.html* del servidor *www.acme.com* al port *8000*:

http://www.acme.com:8000/doc/principal.html

- Primer ens hem de connectar a aquest servidor en aquest port i enviar una petició amb un missatge com aquest:

```
GET /doc/principal.html HTTP/1.0
```

- En aquest cas s'utilitza el mètode GET que vol dir obtenir el recurs especificat.

Missatge resposta (response)

- La **linia inicial** del missatge de resposta rep el nom "***status line***".

versió <SP> codi <SP> frase <CRLF>

- La **versió** especifica el mateix que en el missatge de petició. El **codi** és un nombre de tres dígits que indica el tipus de resposta rebuda d'acord amb el document RFC 7231 secció 6 i la **frase** correspon a un text explicatiu del codi.

Codis resposta

- El codi d'estat d'un missatge de resposta podria ser per exemple:

CODI	SIGNIFICAT
200	Operació efectuada (OK)
301	El recurs sol·licitad ha canviat d'adreça URI (Moved Permanently)
400	Petició incorrecte (Bad Request)
403	Accés prohibit al recurs (Forbidden)
404	No s'ha trobat el recurs (Not Found)
500	Error intern del servidor

- Per més informació: RFC 7231#section-6

Capçaleres HTTP (Headers)

- Els camps que hi pot haver en la capçalera d'un missatge HTTP es poden classificar en un dels següents quatre grups es poden classificar en quatre grups:
  - **Camps de capçalera generals** que poden ser presents tant en els missatges de petició, com en els de resposta.
  - **Camps de capçalera d'entitat** amb informació sobre el cos del missatge o sobre el recurs demanat.
  - **Camps de capçalera de peticions** (RFC 7231 Sec. 5).
  - **Camps de capçalera de respostes** (RFC 7231 Sec. 7) .
  - Llistat de tots els camps d'acord amb RFC 2616.

Alguns exemples de camps generals de capçalera HTTP

- **Date**: indica l'hora i la data en què es va originar el missatge.
- **Pragma**: proporciona una llista de directrius als sistemes que intervenen en la transferència (clients o servidors).
  - **no-cache**: al afegir aquesta directiva a una petició significa que la petició al servidor sempre serà realitzada encara que es trobi una còpia d'aquesta mateixa petició durant el camí entre el client i el servidor.

Alguns exemples de camps generals de capçalera HTTP

- **Cache-Control**: s'utilitza per especificar les directives corresponents als mecanismes de memòries cau al realitzar peticions HTTP.
- **Connection**: permet especificar les opcions de la connexió d'una petició HTTP.
  - **close**: aquesta opció indica que la connexió es tancarà despreś de rebre una resposta a la petició. Els sistemes que no implementen connexions persistents utilitzaran sempre aquesta opció a totes les peticions.

Alguns exemples de camps generals de capçalera HTTP

- **Transfer-Encoding**: codificació aplicada al cos del missatge.
- **Upgrade**: el client informa de les versions del protocol soportades, aleshores el servidor escull la millor.
- **Via**: s'utilitza per part dels servidors intermediaris per afegir informació a la petició com el nom del servidor i el protocol que utilitzen.

Alguns exemples de camps d'entitat de capçalera HTTP

- **Content-Lenght**: indica la longitud en bytes del cos de l'entitat.
- **Content-Type**: indica el tipus de contingut del cos de l'entitat (per exemple text/html, si és un document HTML).
- **Content-Encoding**: indica si s'ha aplicat alguna codificació al cos de l'entitat (per exemple una compressió de dades).
- **Last-Modified**: hora i data en què el recurs contingut en el cos de l'entitat ha sigut modificat per última vegada.

Alguns exemples de camps d'entitat de capçalera HTTP

- **Expires**: Indica la data i l'hora a partir de la qual el contingut del cos es pot considerar obsolet o caducat per al seu emmagatzematge en la memòria cau.
- **Allow**: Indica els mètodes HTTP que es poden aplicar al recurs sol·licitat.
- **Content-Language**: llenguatge utilitzat.
- **Content-Location**: l'URI de l'entitat, en cas que el recurs corresponent en disposi de més d'una. Per exemple un recurs disponible en diferents idiomes.

Alguns exemples de camps d'entitat de capçalera HTTP

- **Content-MD5**: seqüència de bits per a comprovar la integritat del contingut.
- **Content-Range**: per si una entitat s'envia en diferents fragments. Indica el fragment enviat.
- **Etag**: etiqueta associada a l'entitat, per si el recurs en disposa de més d'una. Normalment s'utilitza per identificar els recursos desats a les memòries cau.

Alguns exemples de camps de petició de capçalera HTTP

- **From**: a quest camp conté l'adreça electrònica de l'usuari que sol·licita el recurs.
- **User-Agent**: Aquest camp permet identificar la implementació del client.
- **Referer**: indica en la petició actual l'adreça de la pàgina web (URL) que ha referenciat o enllaçat al recurs demandat. És a dir d'on s'ha obtingut el recurs. El servidor pot utilitzar aquesta informació per generar estadístiques, detectar des de quin recurs es referencia una adreça incorrecta o obsoleta, "hotlinking", etc.

Alguns exemples de camps de petició de capçalera HTTP

- **If-Modified-Since**: Quan el client ja disposa d'una còpia del recurs sol·licitat en la memòria cau, pot utilitzar aquest camp per a dur a terme una operació GET condicional: si el recurs s'ha modificat posteriorment a la data indicada, s'efectuarà l'operació de manera normal i, si no, el servidor enviarà una resposta sense cos i amb el codi 304
- **Authorization**: amb aquest camp, un usuari pot presentar les seves credencials a un servidor perquè li permeti accedir a recursos d'accés restringit.

Alguns exemples de camps de petició de capçalera HTTP

- **Accept**: tipus de contingut que accepta el client.
- **Accept-Charset**
- **Accept-Encoding**
- **Accept-Language**
- **Host**: nom del servidor a qui va dirigida la petició, per si en té més d'un. Permet que un servidor (una sola IP) amb diferents noms actuï com si fossin diferents servidors alhora.

Alguns exemples de camps de petició de capçalera HTTP

- **If-Match**: permet fer una petició condicional d'una entitat partir de proporcionar el llistat de versions que té el client (identificadors d'entitat) de peticions anteriors. S'utilitza per minimitzar el tràfic de xarxa i actualitzar les memòries cau.
- **If-None-Match**
- **If-Unmodified-Since**
- **If-Range**
- **Max-Forwards**
- **Range**: sol·licitar un fragment d'una entitat.

Alguns exemples de camps de petició de capçalera HTTP

- **Max-Forwards**: indica el número de salts que un missatge pot ser redireccionat. En cada salt (proxy o gateway) aquest número disminueix en una unitat, al arribar a 0 el missatge no podrá ser redirigit i no arribará a destí. S'utilitza amb el mètode TRACE.
- **Proxy-Authorization**: amb aquest camp, un usuari pot presentar les seves credencials a un proxy que requereix autenticació.

Alguns exemples de camps de resposta de capçalera HTTP

- **Server**: Aquest camp és similar a User-Agent, però referit al servidor.
- **Location**: Indica que el recurs sol·licitat s'ha d'obtenir en una altra adreça. Aquest serà el cas quan el codi de resposta sigui 301 o 302. Per norma general, quan el client rebi una resposta que inclogui aquest camp, generarà de manera automàtica una nova petició amb l'adreça indicada.

Alguns exemples de camps de resposta de capçalera HTTP

- **WWW-Authenticate**: Aquest camp és obligatori quan el codi de les respostes és 401. Indica que és necessari presentar una credencial per a accedir al recurs sol·licitat, o que la que s'ha presentat no és vàlida.

  Un servidor pot agrupar els seus recursos d'accés restringit en regnes (**realms**), cada un amb el seu esquema d'autenticació i conjunt d'usuaris autoritzats (amb una mateixa credencial s'hauria de poder accedir a tots els recursos d'un regne).

Alguns exemples de camps de resposta de capçalera HTTP

- **Age**: indica el temps que ha passat des de què s'ha generat el recurs servit.
- **Proxy-Authenticate**: el mateix que WWW-Authenticate però per servidors proxy.
- **Public**: llista de mètodes suportats pel servidor.
- **Retry-After**
- **Vary**: llista de camps de la petició que s'han utilitzat per a seleccionar una entitat, quan el recurs en disposa de més d'una.
- **Warning**: informació addicional sobre l'estatus de la resposta.

Mètodes principals HTTP

- El mètode GET**:** serveix per a obtenir l'entitat corresponent a l'URI especificat en la línia de petició. Normalment el servidor traduirà el camí de l'URI a un nom de fitxer o programa:
  - En el primer cas, **el cos de l'entitat serà el contingut del fitxer.**
  - En el segon cas, el servidor executarà el programa i **l'entitat serà el resultat que generi**.
  - Bàsicament és un mètode per obtenir (llegir) un recurs d'un servidor a partir d'una URI i la petició no hauria de tenir efecte sobre les dades.
  - Més informació aquí.

Mètodes principals HTTP

- El mètode HEAD: és igual que el GET, excepte que en la resposta el cos serà buit i, per tant, només tindrà capçalera (que haurà de ser idèntica a la que s'hauria enviat si el mètode fos el GET).
  - Per norma general, s'utilitza el mètode HEAD, per exemple, per a comprovar si un URL és vàlid o per a obtenir informació sobre un recurs sense necessitat de transferir-ne el contingut.
  - Més informació aquí.

Mètodes principals HTTP

- El mètode POST: serveix per a enviar una entitat que el servidor ha d'incorporar en el recurs identificat per l'URI de la línia de petició. La semàntica d'aquest mètode depèn del tipus de recurs. Per exemple, es pot utilitzar per:
  - Afegir contingut a un recurs existent. Per tant, pot modificar un recurs existent.
  - Enviar un missatge a un grup de notícies
  - Crear un registre nou en una base de dades. Per tant, pot crear un nou recurs.
  - Més informació aquí.

Altres Mètodes HTTP

- El mètode PUT: per a crear/actualitzar un recurs amb l'URI especificat en la petició.
- El mètode DELETE: per a esborrar un recurs.
- El mètode OPTIONS: per a obtenir informació sobre les opcions de transferència.
- El mètode TRACE: per a obtenir una còpia del missatge com ha arribat a la seva destinació final.
- Més informació aquí.

Propietats dels mètodes

- **Mètodes segurs**: Mètodes que són bàsicament de només lectura a on el client no espera que es faci cap canvi en l'estat del servidor (per exemple, canvis en el seus recursos). Tot i que el mètode no pot prevenir que s'utilitzi per fer alguna tasca no segura, allò que és important tenir en compte és que el client que va fer la petició no ho demanava i no és responsable de modificacions realitzades a partir de la petició. Són segurs els mètodes GET, HEAD,OPTIONS i TRACE.

Propietats dels mètodes

- **Mètodes Idempotents**: Quan l'efecte de realitzar la mateixa petició sobre el servidor una o multiples vegades causa el mateix efecte que si només s'hagués realitzar una vegada llavors el mètode és idempotent. Igual que en els mètodes segurs, la idempotència no pot preveure el casos en que no s'utilitzi de la manera que estava prevista, en canvi és important que quedi clar quina era la intenció del client quan es va realitzar la petició. Són d'aquest tipus: PUT, DELETE i els mètodes segurs.

Propietats dels mètodes

- **Mètodes ”Cacheables”**: Les respostes del servidor es poden emmagatzemar en el client per ser utilitzades en el futur. Són d'aquest tipus els mètodes GET, HEAD i POST però a la pràctica una grandíssima majoria d'implementacions en servidors i clients del protocols HTTP només permeten emmagatzemar en la memòria cau les respostes a mètodes GET i HEAD, de manera que la resposta a un POST generalment no es pot afegir a la memòria cau.

Autors:

- Pau López
- Ricard Torrell
- Daniel Collados