HTTP

HyperText Transfer Protocol

1

Caratteristiche del protocollo HTTP

- Scambio di messaggi di richiesta e risposta
  - Transazione HTTP o Web
- Protocollo stateless (ciascuna richiesta tra client e server è indipendente dalle altre)
- Basato sul meccanismo di naming degli URI (Uniform Resource Identifier) per identificare le risorse Web
- Metadati sulla risorsa
  - Informazioni sulla risorsa (ma non parte della risorsa) incluse nei trasferimenti; ad esempio:
    - Dimensione della risorsa
    - Tipo della risorsa (ad es. text/html)
      - MIME (Media Type) per classificare il formato dei dati
    - Data dell'ultima modifica della risorsa

2

2

Versioni del protocollo

- HTTP/1.0
  - Versione (quasi) definitiva nel 1996
  - Riferimento: RFC1945 (HTTP/1.0)
  - In precedenza HTTP/0.9 (implementato nel 1990, descritto nel 1992)
- HTTP/1.1
  - Versione definitiva nel 1999
  - Riferimento: RFC2616 (HTTP/1.1)
- HTTP/2
  - È basato su SPDY cioè il protocollo a livello applicativo per il trasporto di contenuti web creato da Google.
  - È stato sviluppato dal Working Group Hypertext Transfer Protocol (httpbis) dell'Internet Engineering Task Force
  - Il Working Group ha presentato HTTP/2 allo IESG(Internet Engineering Steering Group) proponendolo come standard nel dicembre 2014.

3

3

Esempio HTTP 1.0

Il protocollo HTTP/1.0 utilizza una connessione non permanente cioè una connessione che si chiude dopo che il server ha inviato gli oggetti.
Seguiamo i passi per il trasferimento di una pagina Web dal server al client.
Assumiamo che l' indirizzo URL per il file base sia :
www.univpm.it/Entra/Docenti_1/Ingegneria_1.

Ecco ciò che accade:

1. Il client HTTP inizia una connessione TCP a server www.univpm.it.
   La porta numero 80 e usata come numero di default della porta a cui il server HTTP ascolterà i client HTTP che vogliono recuperare i documenti usando l'HTTP.
2. Il client HTTP invia un messaggio di richiesta HTTP al server attraverso il socket associato alla connessione TCP che è stata stabilita in 1.Il messaggio di richiesta include il nome del percorso Entra/Docenti_1/Ingegneria_1.
3. Il server HTTP riceve il messaggio di richiesta attraverso il socket associato alla connessione stabilita al punto 1,trova l' oggetto Entra/Docenti_1/Ingegneria_1 nel sito in cui è immagazzinato(RAM o disco), incapsula l'oggetto all'interno di un messaggio di risposta HTTP e invia il messaggio di risposta al client attraverso il socket.
4. Il server HTTP dice al TCP di concludere la connessione TCP.(Ma il TCP non può chiudere realmente la connessione finchè il client non ha ricevuto il messaggio di risposta intatto.)
5. Il client HTTP riceve il messaggio di risposta. La connessione TCP si conclude. Il messaggio indica che l'oggetto incapsulato è un file HTML. Il client estrae il file dal messaggio di risposta, analizza il file HTML e trova i riferimenti ai professori.
6. L'operazione si ripete fino a che tutti gli oggetti sono scaricati

4

4

Problemi HTTP 1.0

- Per ciascuna di queste connessioni devono essere allocati i buffer del TCP e le variabili del TCP devono essere conservate sia nel client sia nel server.
- Questo può caricare in modo serio il server Web, che potrebbe dover smaltire simultaneamente le richieste di centinaia di client. Secondo ogni oggetto richiesto subisce due RTT ("Round-Trip Time" che è il tempo occorrente a un piccolo pacchetto per viaggiare dal client al server e ritornare al client): uno per stabilire la connessione TCP, l'altro per richiedere e ricevere un oggetto.
- Infine ogni oggetto è sottoposto a una partenza lenta, perché ogni connessione TCP comincia con una fase lenta.

5

5

Problemi HTTP 1.0

- Visto che la dimensione dei file da trasferire va sempre crescendo, la maggior parte dei trasferimenti non va a buon fine, quindi è stato necessario effettuare trasferimenti parziali in modo tale che in caso di interruzione si possa riprendere il download dal punto dove si era interrotto.
- Un altro problema è la diffusione del NON-IP Virtual Hosting, che consiste nel mantenere più server sulla stessa macchina, con un solo indirizzo IP, differenziandoli in base al loro hostname apparente perché di fatto l'indirizzo è solo uno.
- Inoltre, i meccanismi di controllo della cache implementati in questa versione sono rudimentali per cui spesso le applicazioni invece di rischiare di fornire dei dati non aggiornati , non impiegavano la cache, causando l'aumento del traffico di rete.

6

6

HTTP 1.1

- Il protocollo HTTP/1.1 utilizza una connessione permanente.
- Con la connessione permanente, il server lascia aperta la connessione TCP dopo aver spedito la risposta e quindi le successive richieste e risposte fra gli stessi client e server possono essere inviate sull'identica connessione.
- In particolare, un'intera pagina Web può essere spedita su una singola connessione TCP permanente: inoltre, pagine Web multiple residenti sullo stesso server possono essere spedite sulla stessa connessione TCP permanente.

7

7

HTTP 1.1

Esistono due versioni della connessione permanente:

» connessione non incanalata

» connessione incanalata

Nella prima il client passa una nuova richiesta solo quando la risposta alla precedente è stata ricevuta.

In questo caso ciascuno degli oggetti a cui si rimanda riportano un RTT relativo alla richiesta e alla ricezione dell'oggetto.

Sebbene questo sia un miglioramento rispetto ai due RTT della connessione non permanente, il ritardo RTT può essere ancora ridotto con l'incanalamento.

8

8

HTTP 1.1

Un altro svantaggio della mancanza d'incanalamento è che dopo che il server ha spedito un oggetto sulla connessione TCP permanente, la connessione resta agganciata in attesa dell'arrivo di un'altra richiesta; questo comporta spreco delle risorse del server.

Di default l'HTTP/1.1 utilizza la connessione permanente con incanalamento. In questo caso, il client HTTP può fare richieste consecutive(back-to-back)per gli oggetti in riferimento.

Quando il server riceve le richieste può inviare gli oggetti back to back.

Se tutte le richieste e tutte le risposte sono inviate back to back, allora si impiega un solo RTT per tutti gli oggetti in riferimento(invece di un RTT per ogni oggetto in riferimento quando non si usa l'incanalamento).

9

9

Richieste più chiare

Per risolvere i problemi causati dal Virtual Hosting è stato introdotto un nuovo request header; questo header che deve essere incluso in ogni richiesta, specifica quale è il server che intendiamo contattare e su quale porta.

Per esempio se vogliamo contattare il sito di Infomedia sulla porta 8080 ,vediamo come si compone la richiesta:

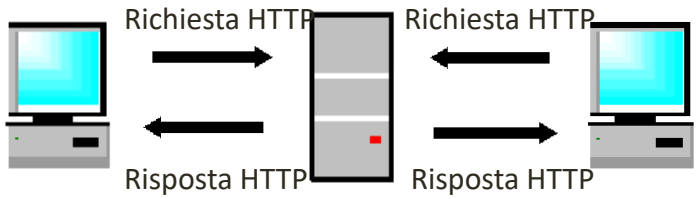
GET / index.html HTTP/1.1
User-agent: Mozilla 3.01C
Accept: image/gif, image/ x- xbi tmap, image/ jpeg,/**
Host: www.univpm.it:8080

Nella prima riga viene specificato l'uso della versione 1.1 del protocollo, mentre nella seconda il request header Host riporta il nome del server che si intende contattare e la porta.

10

10

Comportamento richiesta-risposta dell'HTTP



PC su cui funziona Explorer

Server su cui funziona il server Web NCSA

Mac su cui funziona Navigator

11

11

Messaggi HTTP

- Due tipologie di messaggi:
  - messaggi di richiesta HTTP
  - messaggi di risposta HTTP
- Messaggi composti da:
  - Header o intestazione
    - In formato ASCII
    - Corpo (opzionale)

12

12

Richiesta HTTP

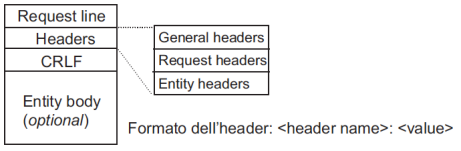
- Una richiesta HTTP comprende
  - Metodo
  - URL
  - Identificativo della versione del protocollo HTTP
  - Ulteriori informazioni addizionali
- Il metodo specifica il tipo di operazione che il client richiede al server
  - GET è il metodo usato più frequentemente: serve per acquisire una risorsa Web
- URL identifica la risorsa locale rispetto al server
- Informazioni addizionali, quali:
  - la data e l'ora di generazione della richiesta
  - il tipo di software utilizzato dal client (user agent)
  - i tipi di dato che il browser è in grado di visualizzare
- … per un totale di oltre 50 tipi di informazioni differenti

13

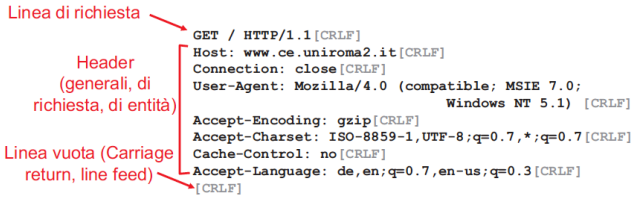
13

Messaggio di richiesta HTTP

- Formato generale



Formato dell'header: <header name>: <value>



14

14

Risposta HTTP

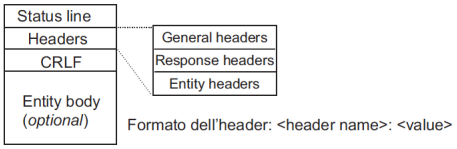
- Una risposta HTTP comprende
  - l'identificativo della versione del protocollo HTTP
  - il codice di stato e l'informazione di stato in forma testuale
  - un insieme di possibili altre informazioni riguardanti la risposta
  - l'eventuale contenuto della risorsa richiesta
- Se la pagina Web richiesta dall'utente è composta da molteplici risorse, ciascuna di esse è identificata da un URL differente: il browser deve inviare un messaggio di richiesta HTTP per ognuna delle risorse incorporate nella pagina

15

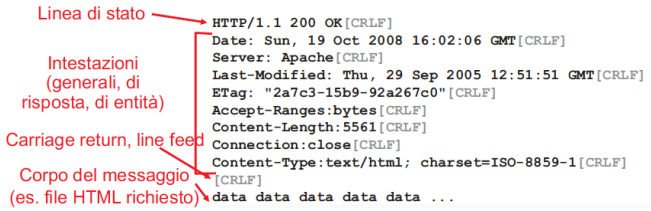
15

Messaggio di risposta HTTP

- Formato generale



Formato dell'header: <header name>: <value>



16

16

I metodi della richiesta

Metodo	Descrizione
GET	Recupera una risorsa dal server (ad es. visitando una pagina)
POST	Invia una risorsa al server (ad. es compilando un modulo)
DELETE	Cancella una risorsa dal server (ad es. eliminando un file)
PUT	Memorizza una risorsa sul server (as es. caricando un file)
HEAD	Recupera solo l'header della risposta senza la risorsa

17

17

I metodi della richiesta

- Un metodo HTTP può essere:
  - sicuro: non altera lo stato della risorsa
  - idempotente: l'effetto di più richieste identiche è lo stesso di quello di una sola richiesta

18

18

Il metodo GET

- E' il metodo più importante e frequente
- Richiede una risorsa ad un server
  - Richiesta composta da solo header (no corpo)
- GET per risorsa statica
  - *GET /foo.html*
- GET per risorsa dinamica (es. risorsa generata con CGI (Common Gateway Interface))
  - *GET /cgi-bin/query?q=foo*
- E' un metodo sicuro ed idempotente
- Può essere:
  - Assoluto: normalmente, ossia quando la risorsa viene richiesta senza altre specificazioni
  - Condizionale: se la risorsa corrisponde ad un criterio indicato negli header If-Match, If-Modified-Since, If-Range, …
  - Parziale: se la risorsa richiesta è una sottoparte di una risorsa memorizzata

19

19

Il metodo HEAD

- Variante di GET usata principalmente per scopi di controllo e debugging
- Il server risponde soltanto con i metadati associati alla risorsa richiesta (solo header), senza inviare il corpo della risorsa
- E' un metodo sicuro ed idempotente
- Usato per verificare:
  - Validità di un URI : la risorsa esiste ed è di lunghezza non nulla
  - Accessibilità di un URI : la risorsa è accessibile presso il server e non sono richieste procedure di autenticazione
  - Coerenza di cache di un URI : la risorsa non è stata modificata rispetto a quella in cache, non ha cambiato lunghezza, valore hash o data di modifica

20

20

Il metodo POST

- Permette di trasmettere delle informazioni dal client al server
  - Aggiornare una risorsa esistente o fornire dati di ingresso
  - Dati forniti nel corpo della richiesta (GET: dati codificati nell'URI di richiesta)
- Ad es. usato per sottomettere i dati di un form HTML ad un'applicazione sul server identificata dall'URI specificata nella richiesta
- E' un metodo né sicuro, né idempotente
- Il server può rispondere positivamente in tre modi:
  - *200 OK* : dati ricevuti e sottomessi alla risorsa specificata; è stata data risposta
  - *201 Created* : dati ricevuti, la risorsa non esisteva ed è stata creata
  - *204 No content*: dati ricevuti e sottomessi alla risorsa specificata; non è stata data risposta

21

Il metodo PUT

- Serve per trasmettere delle informazioni dal client al server, creando o sostituendo la risorsa specificata
- Differenza tra PUT e POST:
  - l'URI di POST identifica la risorsa che gestirà l'informazione inviata nel corpo della richiesta
  - l'URI di PUT identifica la risorsa inviata nel corpo della richiesta: è la risorsa che ci si aspetta di ottenere facendo un GET in seguito con lo stesso URI
- Non è sicuro, ma è idempotente
- Non offre nessuna garanzia di controllo degli accessi o locking
  - Estensione WebDAV (Web-based Distributed Authoring and Versioning) del protocollo HTTP: fornisce (tra le altre cose) una semantica sicura e collaborativa per il metodo PUT

22

Header HTTP

- Gli header sono righe testuali free-format che specificano caratteristiche:
  - generali della trasmissione (header generali)
  - dell'entità trasmessa (header di entità)
  - della richiesta effettuata (header di richiesta)
  - della risposta generata (header di risposta)
- Formato dell'header:
  - *<header name>: <value>*

23

23

Header HTTP (2)

- Header generali
  - Si applicano solo al messaggio trasmesso e si applicano sia ad una richiesta che ad una risposta, ma non necessariamente alla risorsa trasmessa
  - Ad es., *Date*: per data ed ora della trasmissione
  - Ad es., Pragma: no-cache per risposta direttamente dall'origin server (no copia in cache di qualche proxy)
- Header di entità
  - Forniscono informazioni sul corpo del messaggio, o, se non vi è corpo, sulla risorsa specificata
  - Ad es., *Content-Type*: il tipo MIME dell'entità acclusa
- Header obbligatorio in ogni messaggio che abbia un corpo
  - Ad es., *Content-Length*: la lunghezza in byte del corpo

24

24

Header HTTP (3)

- Header di richiesta
  - Impostati dal client per specificare informazioni sulla richiesta e su se stesso al server
  - Ad es., *User-Agent*: stringa che descrive il client che origina la richiesta; tipicamente: tipo, versione e sistema operativo del client
  - Ad es., *Referer*: l'URL di provenienza (utile per user profiling e debugging)
- Header di risposta
  - Impostati dal server per specificare informazioni sulla risposta e su se stesso al client
  - Ad es., Server: stringa che descrive il server che origina la risposta; tipicamente: tipo, versione e sistema operativo del server

25

25

Codice di stato della risposta

- E' un numero di tre cifre, di cui la prima indica la classe della risposta e le altre due la risposta specifica
- Esistono le seguenti classi:
  - 1xx: Informational una risposta temporanea alla richiesta, durante il suo svolgimento
  - 2xx: Successful il server ha ricevuto, capito e accettato la richiesta
  - 3xx: Redirection il server ha ricevuto e capito la richiesta, ma possono essere necessarie altre azioni da parte del client per portare a termine la richiesta
  - 4xx: Client error la richiesta del client non può essere soddisfatta per un errore da parte del client (errore sintattico o richiesta non autorizzata)
  - 5xx: Server error la richiesta può anche essere corretta, ma il server non è in grado di soddisfare la richiesta per un problema interno (suo o di applicazioni invocate per generare dinamicamente risorse)

26

26

Alcuni codici di stato della risposta

- 200 OK
  - Risorsa nel corpo del messaggio
- 301 Moved Permanently
  - Redirezione: risorsa spostata
- 304 Not Modified
  - Risorsa non modificata
- 401 Unauthorized
  - La risorsa richiede autenticazione dell'utente
- 403 Forbidden
  - Accesso vietato
- 404 Not Found
  - Risorsa non esistente
- 500 Internal Server Error
  - Errore imprevisto che impedisce il servizio richiesto
- 501 Not Implemented
  - Il server non supporta la funzionalità richiesta (es. Metodo non implementato)

27

27

Connessione (1)

- Introduzione di connessioni persistenti e pipelining (La connessione TCP è non persistente)
- Connessione persistente: possibilità di trasferire coppie multiple di richiesta e risposta in una stessa connessione TCP
- Vantaggi delle connessioni persistenti
  - Riduzione dei costi (instaurazione ed abbattimento) delle connessioni TCP
- 3-way handshake del TCP: solo per instaurare la connessione iniziale
  - Riduzione della latenza
- Aumenta la dimensione della finestra di congestione del TCP
  - Controllo di congestione a regime
  - Migliore gestione degli errori
- Problemi delle connessioni persistenti
  - Concorrenza minore
  - Quando chiudere la connessione TCP?

28

28

Connessione (2)

- Pipelining: trasmissione di più richieste senza attendere l'arrivo della risposta alle richieste precedenti
- Le risposte devono essere fornite dal server nello stesso ordine in cui sono state fatte le richieste
  - HTTP non fornisce un meccanismo di riordinamento esplicito
  - Il server può tuttavia processare le richieste in un ordine diverso da quello di ricezione
- Vantaggi del pipelining
  - Ulteriore riduzione del tempo di latenza e ottimizzazione del traffico di rete
- Soprattutto per richieste che riguardano risorse molto diverse per dimensioni o tempi di elaborazione

29

29

Chiusura di connessioni persistenti

- Header generale usato per segnalare la chiusura della connessione persistente

  Connection: close
- L'implementazione della chiusura non è specificata nel protocollo HTTP
  - Può essere iniziata dal client o dal server
- Il server può chiudere la connessione:
  - allo scadere di un timeout, applicato sul tempo totale di connessione o sul tempo di inattività di una connessione
  - allo scadere di un numero massimo di richieste da servire sulla stessa connessione

30

30

Hosting virtuale

- Ad uno stesso IP possono corrispondere nomi diversi e server diversi
  - Requisito importante per i "provider"
  - IP e porta non sono più sufficienti ad identificare il server
- Header di richiesta Host in HTTP
  - Serve a specificare il nome (e la porta) del server

  Host: www.univpm.it
  - E' obbligatorio
  - Permette l'implementazione del virtual hosting senza manipolazioni del routing e multi-addressing IP

31

31

Negoziazione del contenuto

- Header di richiesta HTTP/1.1 per la negoziazione del contenuto
  - Accept, Accept-Charset, Accept-Encoding, Accept-Language, TE
- Esigenza del client di negoziare con il server la rappresentazione preferita di una risorsa
  - Accept: tipo di media preferito
  - Accept-Charset: insiemi di caratteri preferiti
  - Accept-Encoding: codifica del contenuto preferita
  - Accept-Language: linguaggio preferito
  - TE: codifica del trasferimento preferita
- Modalità di negoziazione del contenuto:
  - Guidata dal client: il client sceglie tra le alternative possibili ed indica la propria preferenza al server
  - Guidata dal server: il server sceglie la rappresentazione della risorsa in base alle informazioni ricevute dal client

32

32

Autenticazione digest in HTTP

- Il server invia al browser una stringa generata in modo univoco
  - *"nonce"*
- Il browser risponde con
  - nome utente
  - un valore crittografato basato su: nome utente, password, URI e nonce (digest MD5 di 128 bit)
- Il browser ricorda l'autorizzazione
- Vantaggi
  - La password non viene trasmessa in chiaro
  - Meccanismo più sicuro rispetto all'autenticazione Basic
- Ma il meccanismo non è sicuro al 100%
  - E' possibile intercettare la richiesta con URI, nonce e response e riprodurla per accedere alle risorse protette

33

33

Cookie (1)

I cookie sono un meccanismo alternativo che possono usare i siti per tener traccia degli utenti.

Vediamo un esempio.

Supponete che un cliente contatti un sito Web per la prima volta, e che per questo sito usi i cookie. La risposta del server includerà una intestazione Set-cookie: Spesso questa linea di intestazione contiene un numero di identificazione generato dal server Web.

Per esempio la linea di intestazione può essere:

Set-cookie: 1678453

34

34

Cookie (2)

Quando il client HTTP riceve il messaggio di risposta vede la linea di intestazione Set-cookie: e il numero di identificazione. Esso allora appende una linea a uno speciale file cookie che è immagazzinato nella macchina client. Questa linea di solito comprende il nome dell'host del server associato al numero di identificazione dell'utente. Nelle richieste successive allo stesso server, diciamo una settimana dopo, il client include un'intestazione di richiesta Cookie: e questa intestazione specifica il numero di identificazione di quel server. Nell'esempio corrente, il messaggio di richiesta comprende la linea di intestazione:

Cookie: 1678453

35

35

LA CACHE DEL WEB (1)

- La cache del Web(detta anche proxy server) è un entità della rete che soddisfa le richieste HTTP da parte di un client.
- La cache del Web ha un proprio disco di archiviazione e conserva nella sua memoria copie degli oggetti richiesti di recente.
- Gli utenti configurano i loro browser cosicché tutte le loro richieste HTTP siano prima dirette alla cache.
- Una volta configurato un browser, ciascuna richiesta del browser per un oggetto è prima indirizzata alla cache.

36

36

LA CACHE DEL WEB (2)

- Per esempio supponiamo che il browser stia chiedendo l'oggetto http//www.scuola.edu / campus.gif.
- •il browser stabilisce una connessione TCP con la cache del Web e invia una richiesta HTTP per l'oggetto alla cache del Web.
- • la cache controlla se ha una copia dell'oggetto memorizzata localmente.
- Se c'è, la cache inoltra l'oggetto all'interno di un messaggio di risposta HTTP al browser del client.
- • se la cache non ha l'oggetto, apre una connessione TCP al server di origine, cioè, a www.scuola.edu.
- La cache invia allora una richiesta HTTP per l'oggetto nella connessione TCP. Dopo aver ricevuto questa richiesta, il server originale invia l'oggetto all'interno di una risposta HTTP alla cache.

37

37

LA CACHE DEL WEB (3)

Quando la cache riceve l'oggetto, ne archivia una copia nella memoria locale e invia la copia, all'interno di un messaggio di risposta HTTP, al browser del client.

- La cache è sia un server sia un client allo stesso tempo
  - Quando riceve le richieste e invia le risposte a un browser è un server.
  - Quando invia le richieste e riceve le risposte è un client.
- Le cache del Web hanno un grande successo in Internet per almeno tre ragioni.
  - Primo, una cache può sostanzialmente ridurre il tempo di risposta a una richiesta del client soprattutto se il collo di bottiglia della larghezza di banda fra il client e il server di origine è inferiore a quello tra il client e la cache.
  - Secondo, la cache può ridurre il traffico su un link di accesso Internet di una istituzione. Attraverso la riduzione del traffico, l'istituzione(per esempio una società o un'università) non deve rinnovare la larghezza di banda così spesso, e quindi riduce i costi.
  - Inoltre, la cache può ridurre in modo consistente il traffico dell'intero Internet, migliorando quindi le performance di tutte le applicazioni.

38

38

La gestione della cache

Il campo dove si sono compiuti gli sforzi maggiori nella definizione dell'HTTP versione 1.1 è la gestione della cache.

L'obiettivo è quello di ridurre al minimo il traffico di rete e garantire la sicurezza del contenuto della cache, in modo tale da consentirne l'uso alle applicazioni senza che l'utente rischi di avere informazioni non aggiornate.

Il primo passo è stato definire la validità di ogni risorsa, grazie all'uso di un nuovo response header:

Expires: Thu, 01 Dec 2020 16:00:00 GMT

39

39

La gestione della cache

Il response header dà un indicazione diretta della data oltre la quale il dato in cache non sarà più affidabile.

Prendiamo come esempio la seguente richiesta:

GET /index.html HTTP/ 1.1
User-Agent: Mozilla 3. 01C
Accept: image/ gif , image / x-xbitmap ,image/ jpeg, */*
Host: www.infomedia.it
If-modified-since: Thu , 01 Dec 2020 16: 00: 00 GMT

Supponiamo che la pagina sul server sia stata modificata il primo gennaio del 2018 ma che la richiesta passi tramite un proxy che ne ha una versione aggiornata al primo gennaio 2019.

40

40

La gestione della cache

Mentre con la versione 1.0 avremmo ricevuto la copia del proxy, con la versione 1.1 quello che riceveremo dipenderà dalla data di scadenza della pagina: se la data lascia una settimana di validità , il proxy si accorgerà di avere una copia ormai scaduta e la richiederà automaticamente al server.

Ricevuta la pagina aggiornata la memorizzerà nella sua cache e ce la invierà con un header di questo tipo:

HTTP/ 1.1 200 OK
Date: Mon, 11 May 1998 15:35:12 GMT
Server:Apache/ 1.2.4
Last-Modified: Sun, 10 May 2018 18:11:27 GMT
Expires: Mon, 18 May 2018 15:35:12 GMT

Content-Length: 16225
Content-Type: text/ html

Con questo header ci indica che la pagina sarà valida ancora per sette giorni.

41

41

Caching cooperativo

Utilizzando cache Web multiple, situate in diversi posti in Internet, è possibile cooperare e migliorare le performance generali.

Per esempio, la cache di una istituzione può essere configurata per inviare le sue richieste HTTP a una cache in una colonna portante di ISP a livello nazionale. In questo caso, quando la cache dell'istituzione non contiene l'oggetto richiesto, essa invia la richiesta HTTP alla cache nazionale.

La cache nazionale invia quindi l'oggetto (all'interno di un messaggio di risposta HTTP) alla cache dell'istituzione, che a sua volta lo inoltra al browser richiedente.

Il vantaggio di passare attraverso una cache di più alto livello, come una nazionale, è che ha molti più utilizzatori e quindi maggiori hit rate.

42

42

HTTPS

- La soluzione più sicura: HTTPS
- HTTPS trasmette i dati in HTTP semplice su un protocollo di trasporto (SSL) che crittografa tutti i pacchetti
  - Il server ascolta su una porta diversa (per default la porta 443) e si usa uno schema di URI diverso (introdotto da https://)
- SSL: Secure Socket Layer
  - Crittografia a chiave pubblica (certificato)
  - Trasparente

43

43

Client da installare

- https://www.getpostman.com/downloads/

44

44

45