Titolo

- HTTP

Caratteristiche del protocollo HTTP

- • Scambio di messaggi di richiesta e risposta
  – Transazione HTTP o Web
- • Protocollo stateless
- • Basato sul meccanismo di naming degli URI 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 per classificare il formato dei dati
    - • Data dell'ultima modifica della risorsa

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 (quasi) definitiva nel 1999
  - Riferimento: RFC2616 (HTTP/1.1)

Messaggi HTTP

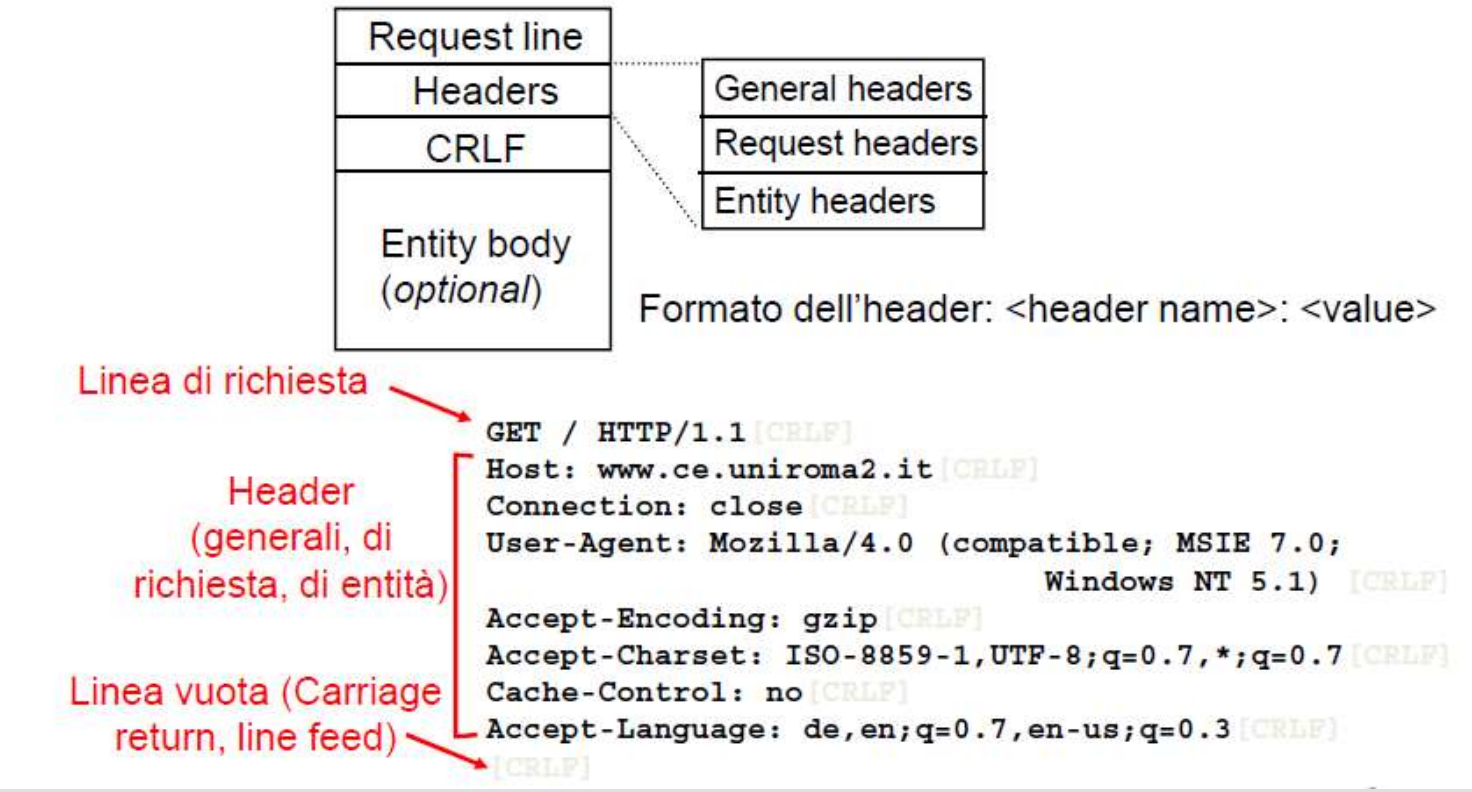
- • Due tipologie di messaggi:
  – messaggi di **richiesta** HTTP
  – messaggi di **risposta** HTTP
- • Messaggi composti da:
  – Header o intestazione
    In formato ASCII (leggibile dagli esseri umani)
  – Corpo (*opzionale*)

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

Messaggio di richiesta HTTP

- Formato generale



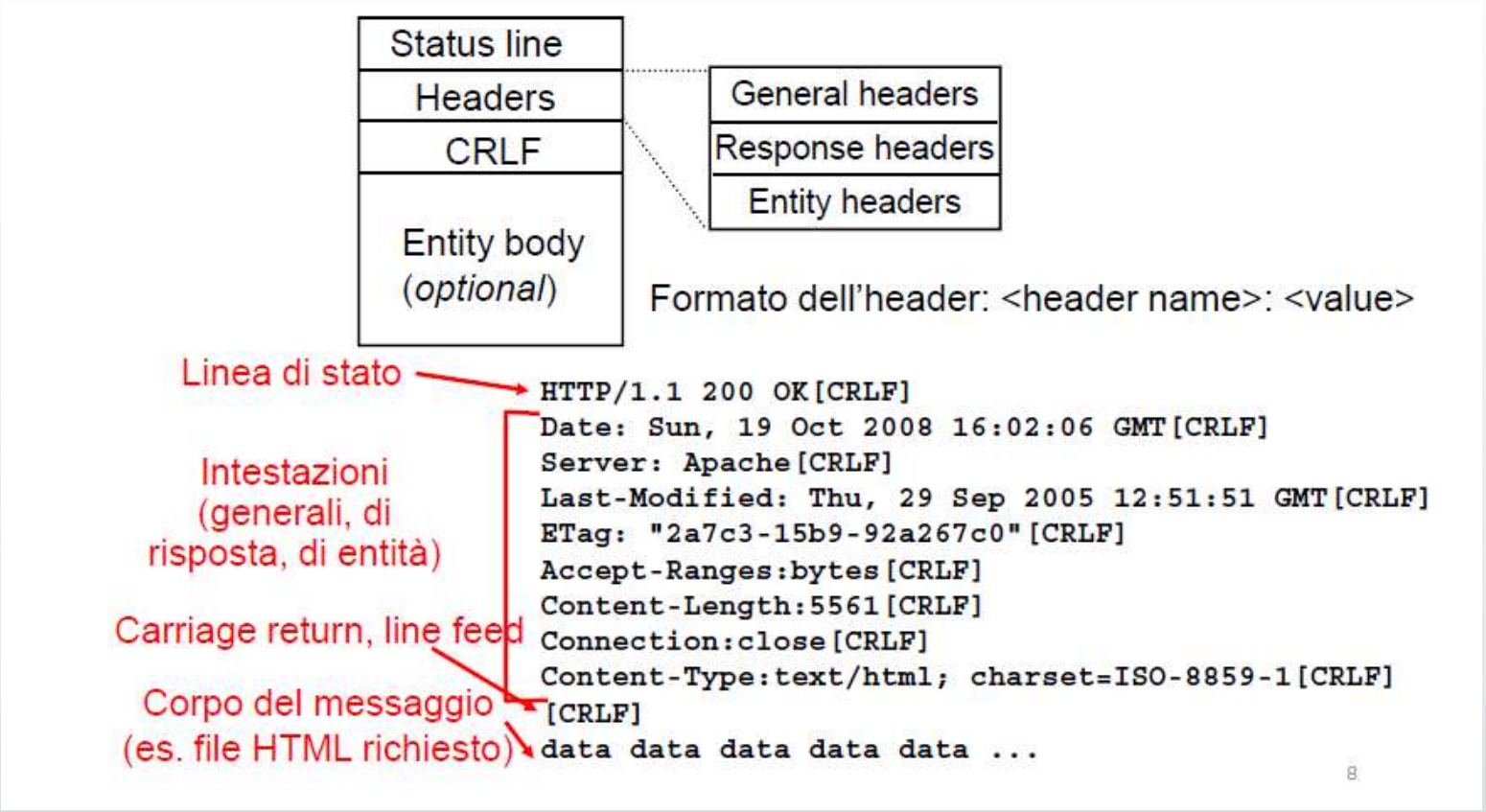
Risposta HTTP

- 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

Messaggio di risposta HTTP

- Formato generale



Metodi della richiesta

Metodo	Versione HTTP	Modifiche in HTTP/1.1
GET	1.0 e 1.1	Richiesta di parti di entità
HEAD	1.0 e 1.1	Uso di header di richiesta condizionali
POST	1.0 e 1.1	Gestione di connessione, trasmissione di messaggio
PUT	1.1 (1.0 non standard)	
DELETE	1.1 (1.0 non standard)	
OPTIONS	1.1	Estensibilità
TRACE	1.1	Estensibilità
CONNECT	1.1	Uso futuro

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)
  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

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

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
  – 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 conten**t: dati ricevuti e sottomessi alla risorsa specificata; non è stata data risposta

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
- Riferimento: RFC 2518

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>

Header HTTP

- 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
  • RFC 1123 per il formato della data (possibili anche altri formati)
  – 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

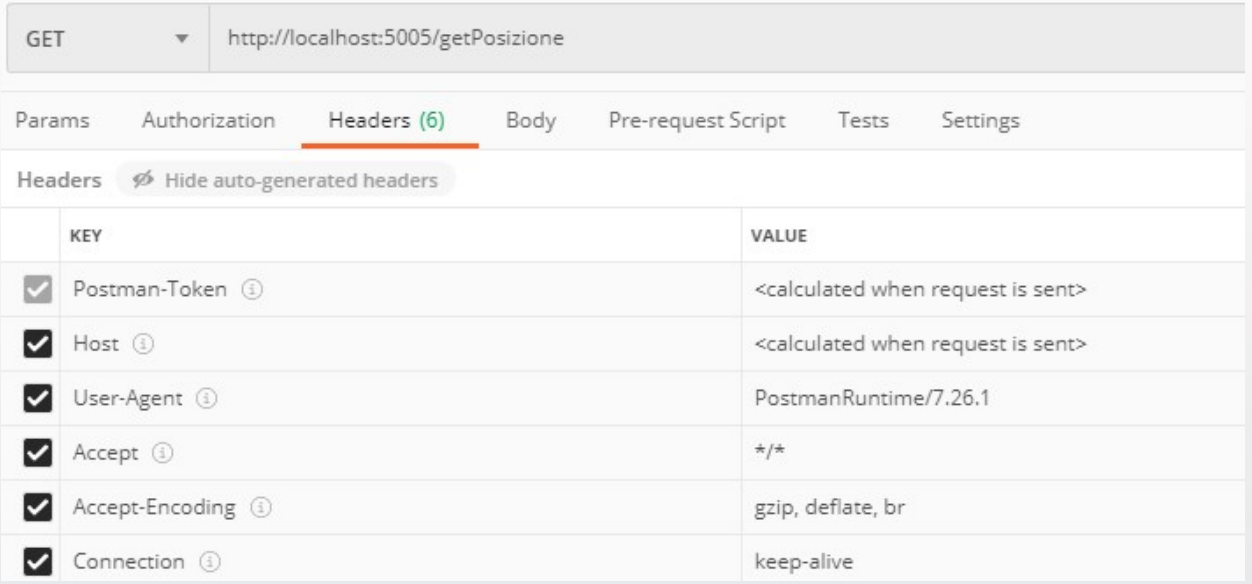
Header HTTP

- 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

Header di richiesta in HTTP/1.0

- 5 header di richiesta in HTTP/1.0
- **From:**
  gorby@moskvar.com
- **User-Agent:**
  esempio: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
- **If-Modified-Since:**
  Thu, 01 Apr 2002 16:00:00 GMT
- **Referer:**
  http://www.ce.uniroma2.it/index.html
- **Authorization:**
  Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==



Header di risposta in HTTP/1.0

- 3 header di risposta in HTTP/1.0
- **Server:**
  Apache 1.3.20
- **Location:**
  http://www.uniroma2.it/newindex.html
- **WWW-Authenticate:**
  Basic realm="Area Privata"

Header di entità in HTTP/1.0

- Sei header di entità in HTTP/1.0
- **Content-Type:** application/json
- **Content-Length:** 6

Body Cookies Headers (5) Test Results Status: 200 OK

KEY	VALUE
Content-Type	text/html; charset=utf-8
Content-Length	6
Access-Control-Allow-Origin	*
Server	Werkzeug/0.16.0 Python/3.6.2
Date	Wed, 08 Jul 2020 08:33:16 GMT

- **Content-Encoding:**
  – Tipo di compressione applicato alla risorsa (esempio gzip)
- **Allow:** GET
  – Metodi supportati dalla risorsa specificata dall'URI
- **Last-Modified:**
  Tue, 26 Mar 2019 16:52:23 GMT
- **Expires:**
  - Data di scadenza

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)

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)

Fine

Grazie