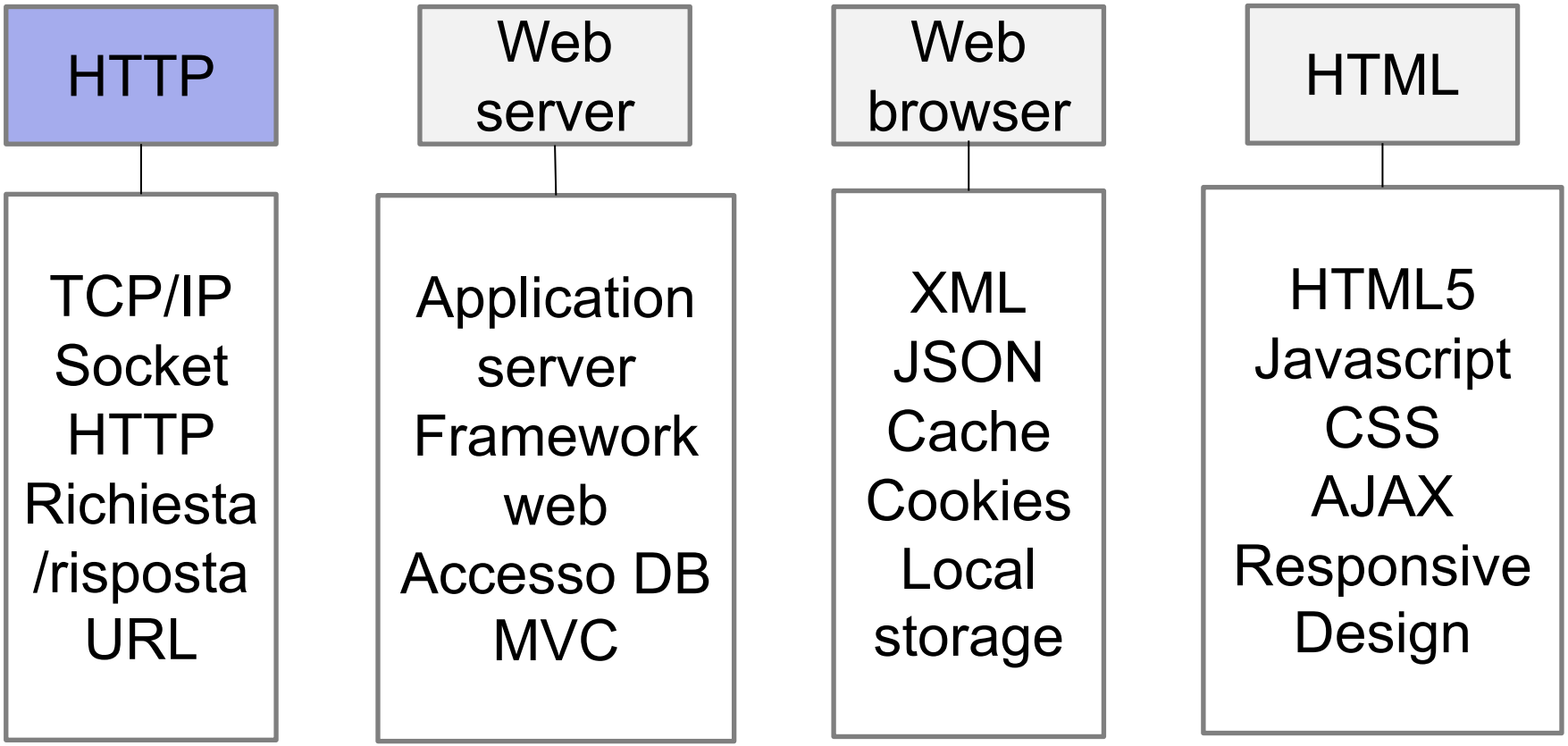
SUPSI

Protocollo HTTP

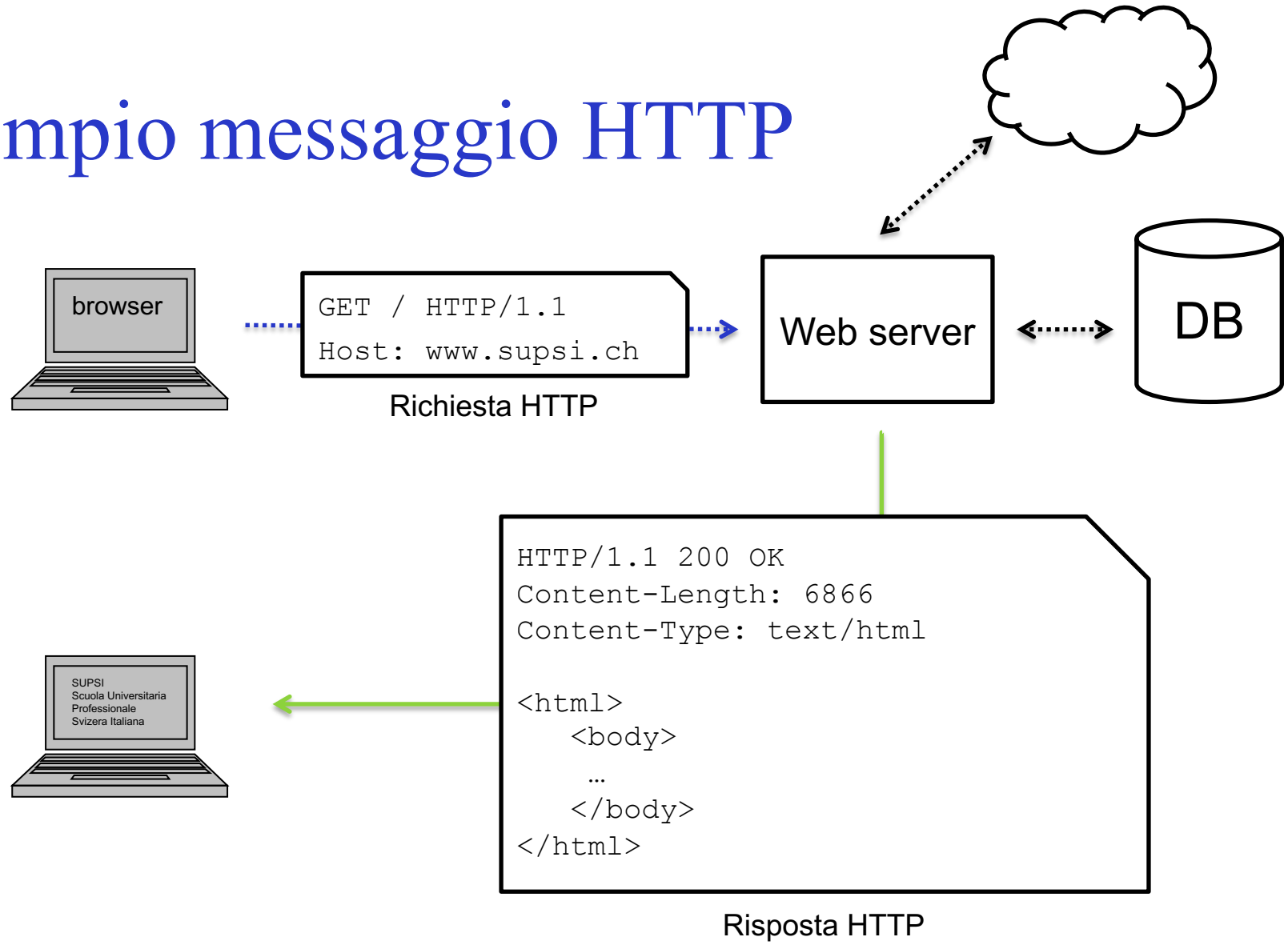
Componenti di una web app



Protocollo HTTP

- L'**HyperText Transfer Protocol** (HTTP) (protocollo di trasferimento di un ipertesto) è usato come principale sistema per la trasmissione d'informazioni sul web ovvero in un'architettura tipica client-server.
- La sua porta di default è la 80
- Un **ipertesto** è formato da un insieme di documenti, collegati tra loro tramite riferimenti ipertestuali, denominati **link**.
- Meccanismo **richiesta/risposta.** I messaggi HTTP sono difatti di 2 tipi: request e response
- Si colloca nello strato Applicativo al di sopra di **TCP/IP**
- Le specifiche del protocollo sono gestite dal **World Wide Web Consortium** (W3C). Ad oggi versione più utilizzata è la 1.1.

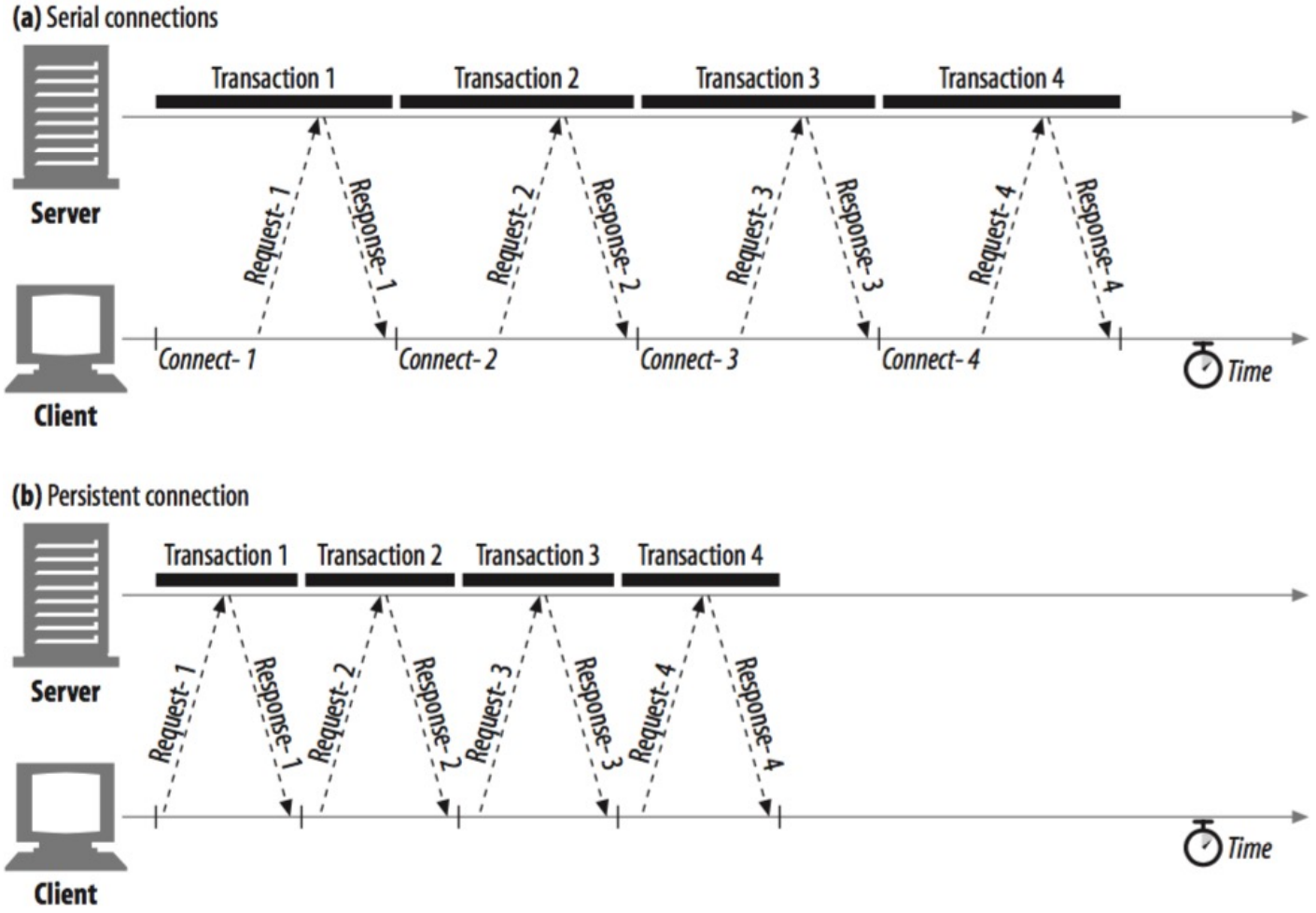
Esempio messaggio HTTP



Esempio messaggio HTTP

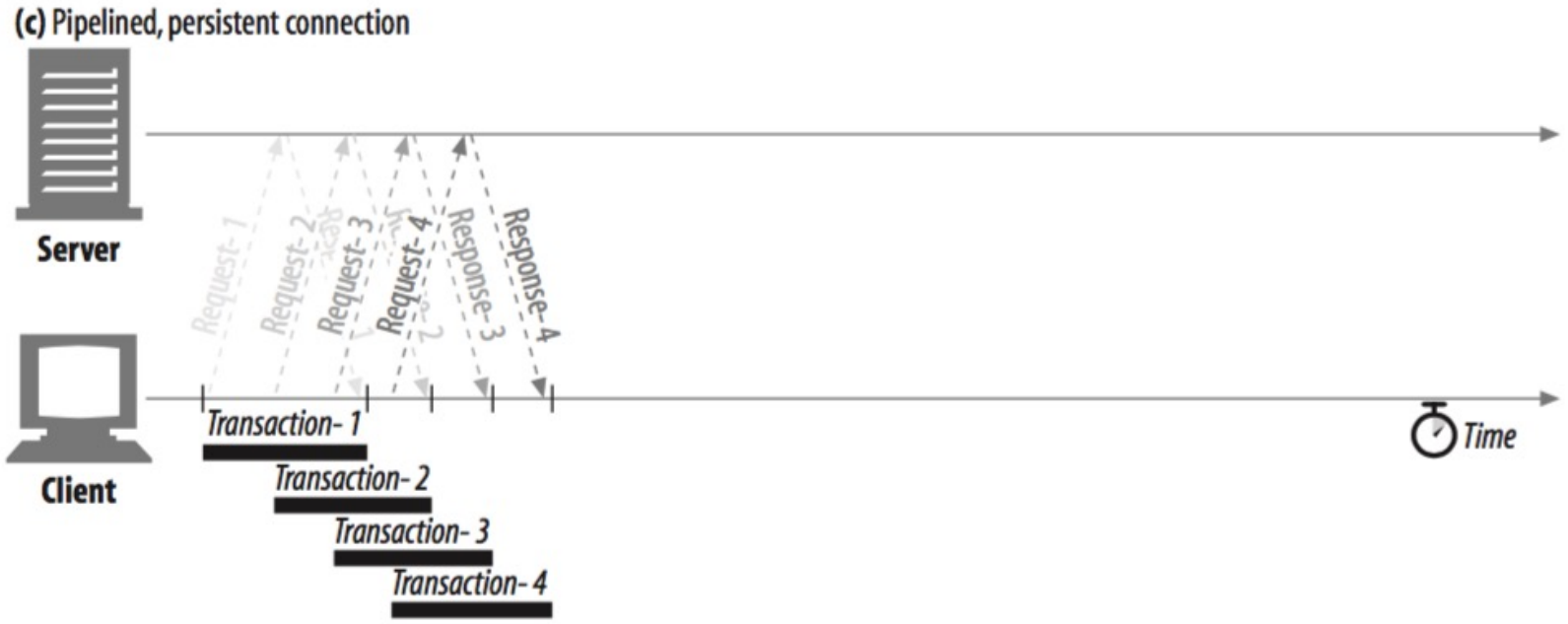
- Il browser del client invia al server una richiesta HTTP, specificando l'indirizzo (URL) della risorsa su una determinta porta (la 80 di default)
- Il server, che è in ascolto su quella porta, esegue le seguenti operazioni:
  1. apre una connessione con il client per servire la richiesta con l'azione corretta (metodo: `GET, POST, HEAD, …`)
  2. invia una struttura dati chiamata risposta HTTP, contenente l'esito della richiesta, ovvero normalmente il file che il browser visualizzerà
  3. a partire dalla versione 1.1, processa ulteriori richieste del medesimo client (in 1.0, ad ogni richiesta/risposta corrisponde una nuova connessione)
  4. chiude la connessione

Da HTTP 1 a HTTP 1.1



Fonte: **HTTP: The Definitive Guide**

Fino a pipelined connections



Fonte: **HTTP: The Definitive Guide**

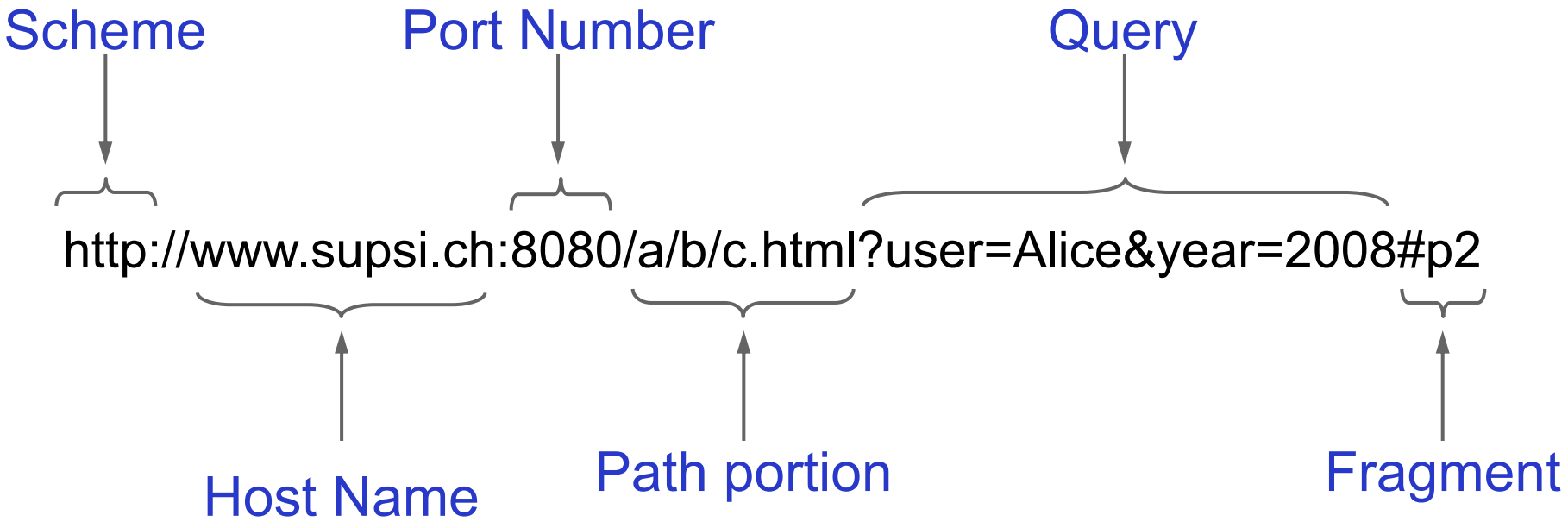
URL (Uniform Resource Locator)

- Le risorse web sono identificate da una particolare stringa, detta URL, la cui forma è:

```
http://<hostname>:<porta>/<percorso-risorsa>?<query>#<frammento>
```

  - hostname: nome o indirizzo IP del server che ospita la risorsa
  - porta: socket su cui il server è in ascolto (opzionale, la porta associata di default al protocollo HTTP è la 80)
  - percorso-risorsa: percorso della risorsa all'interno del server
  - query: stringa passata alla risorsa per ulteriore interpretazione (opzionale)
  - frammento: identifica una parte della risorsa stessa (opzionale)

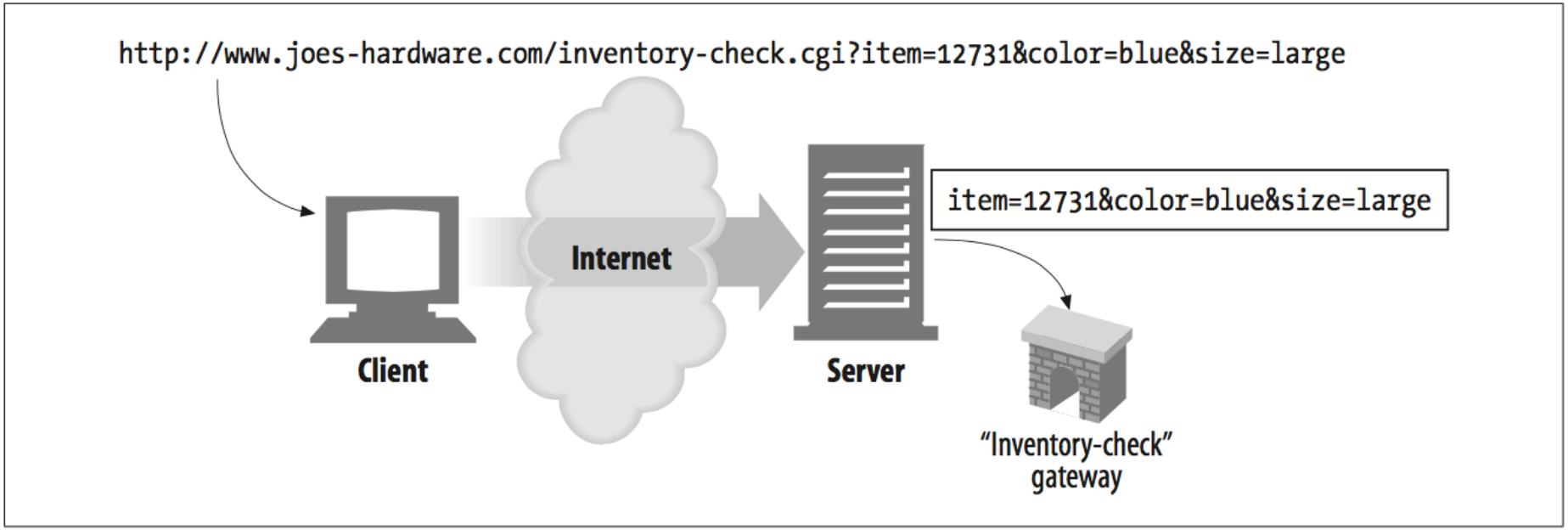
URL


Scheme
Port Number
Query
http://www.supsi.ch:8080/a/b/c.html?user=Alice&year=2008#p2
Host Name
Path portion
Fragment

Esempi di URL

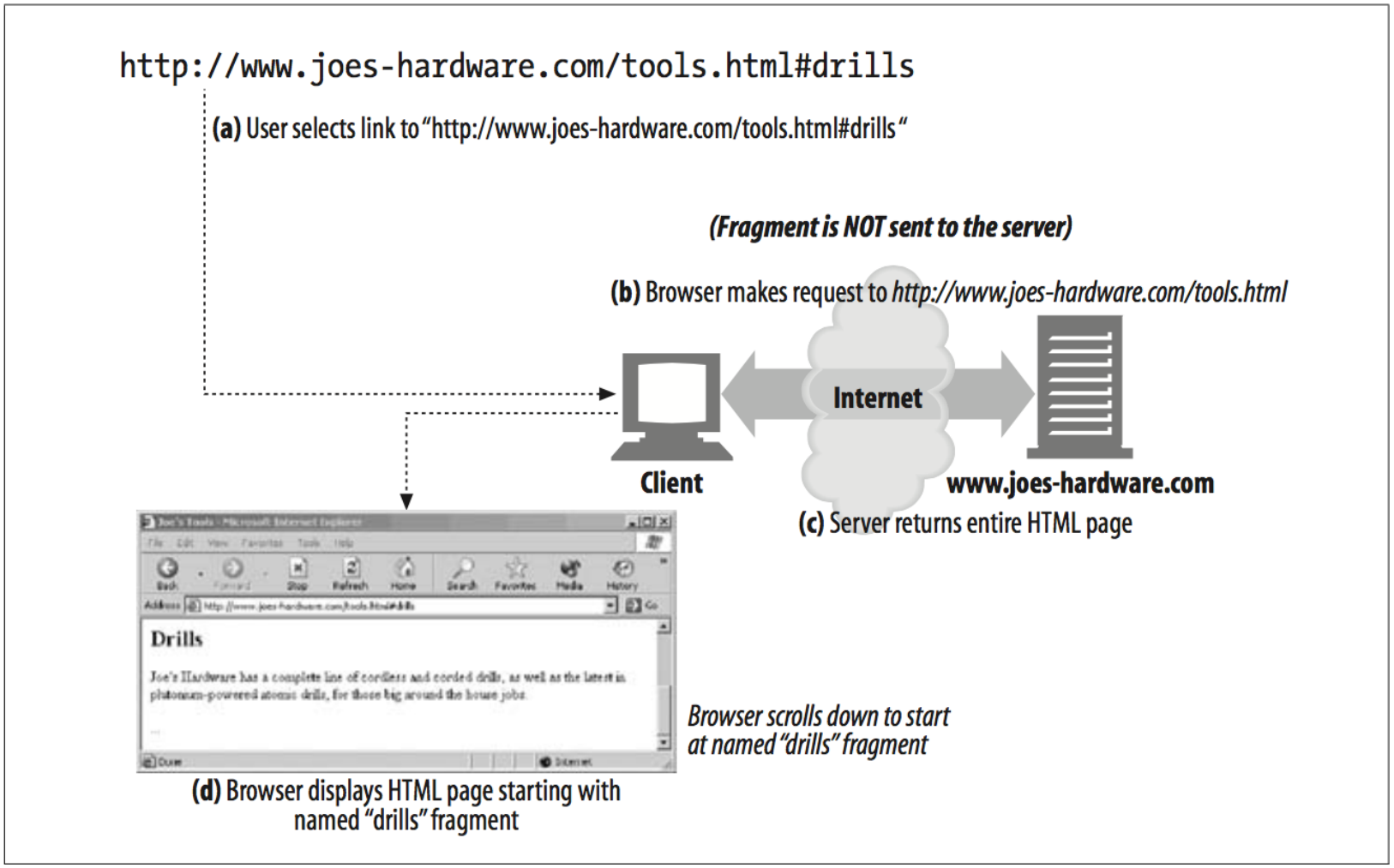
- http://www.supsi.ch
- http://localhost**:8080**/index.html
- https://www.google.ch/search**?q=supsi&hl=it**
- https://it.wikipedia.org/wiki/Scuola_universitaria_professionale_della_Svizzera_italiana#Struttura

Query string



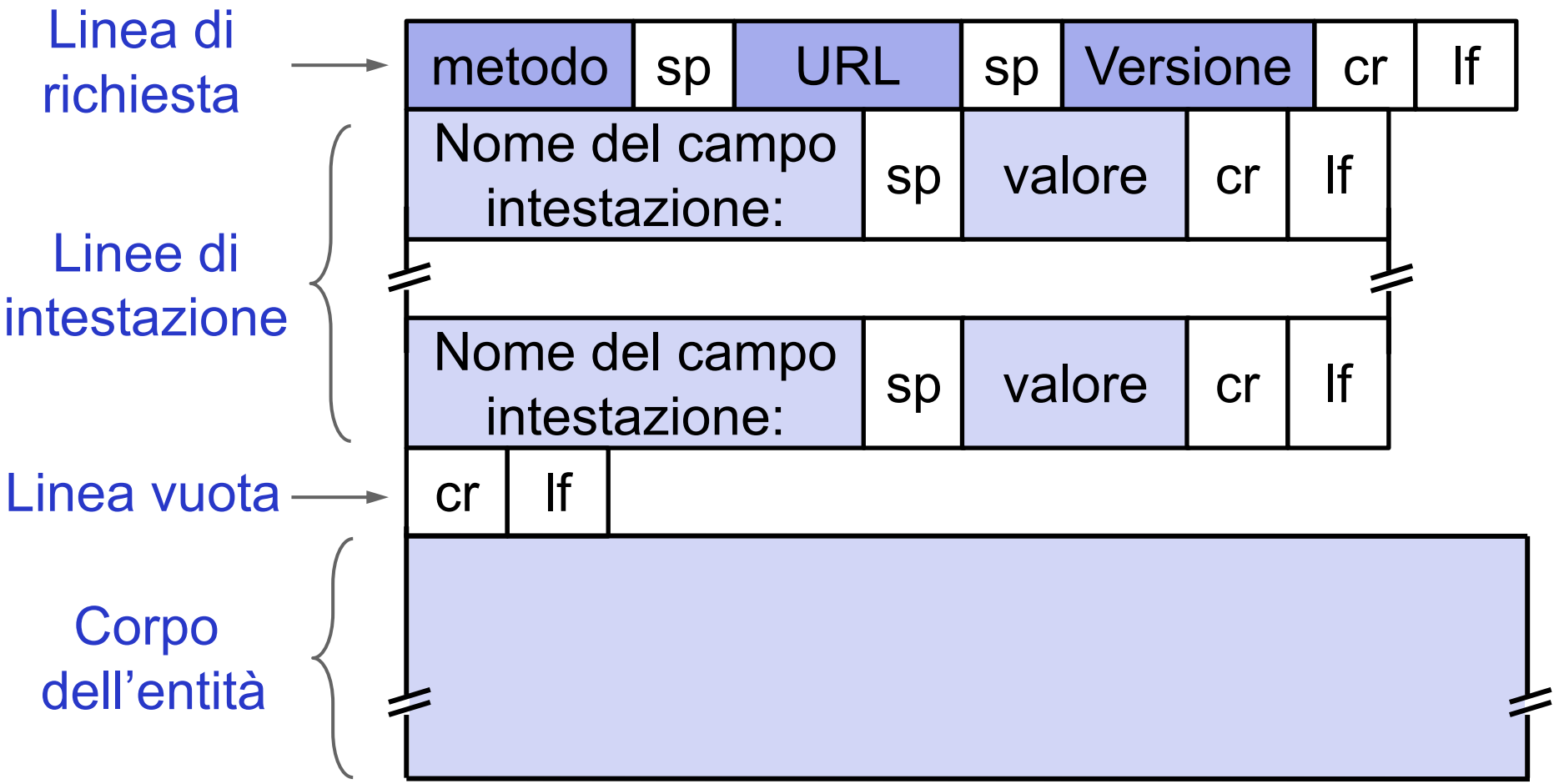
Fonte: **HTTP: The Definitive Guide**

Fragment

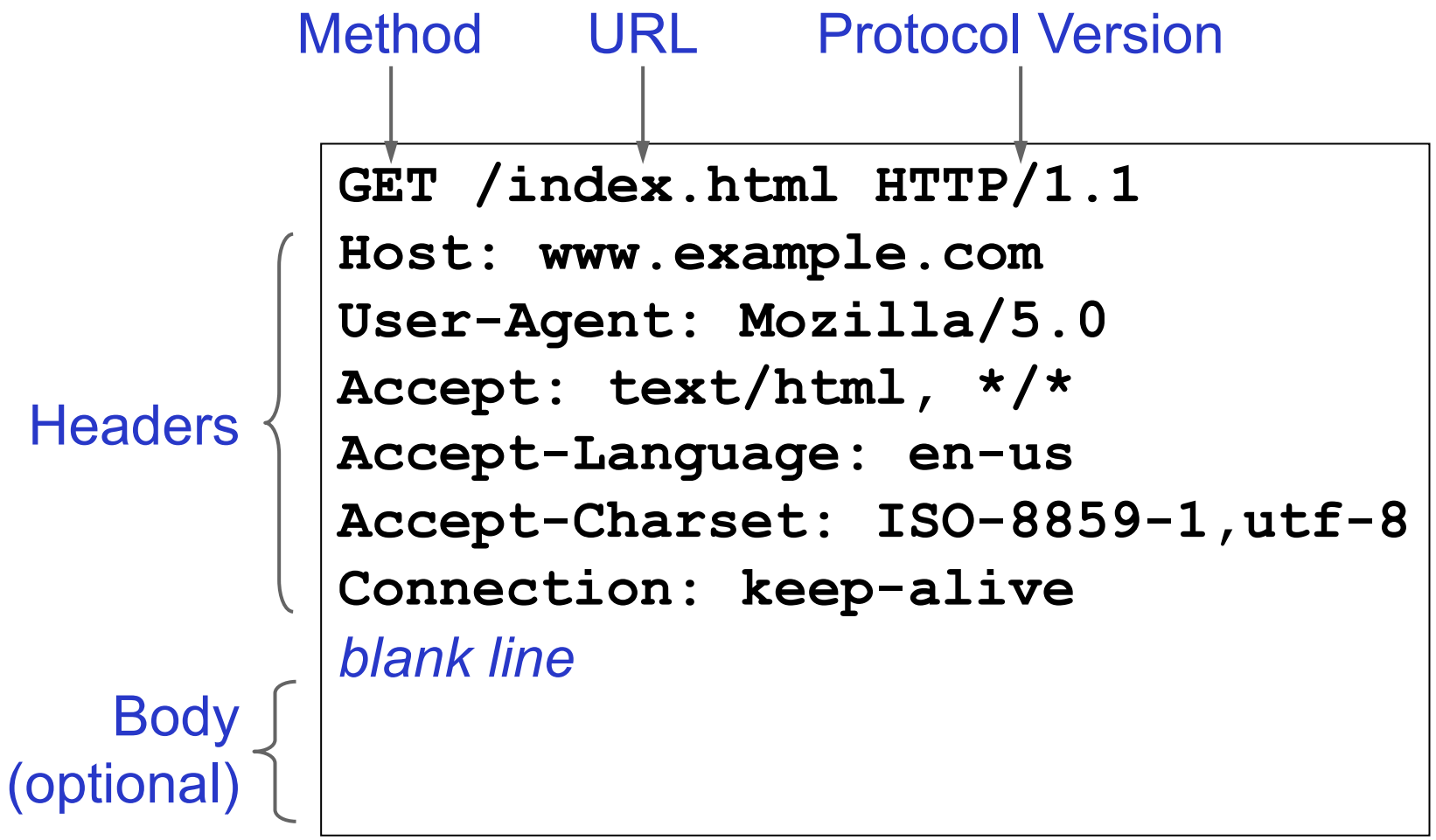


Fonte: **HTTP: The Definitive Guide**

Formato generale richiesta HTTP



Esempio richiesta HTTP



Header di richiesta

- Gli header di richiesta più comuni sono:
  - **Host**: nome del server a cui si riferisce l'URL. È obbligatorio nelle richieste conformi HTTP/1.1 perché permette l'uso dei virtual host basati sui nomi.
  - **User-Agent**: identificazione del tipo di client: tipo browser, produttore, versione...
  - altri: **Referer**, **Accept**, **Authorization**, **Cookie, Connection**

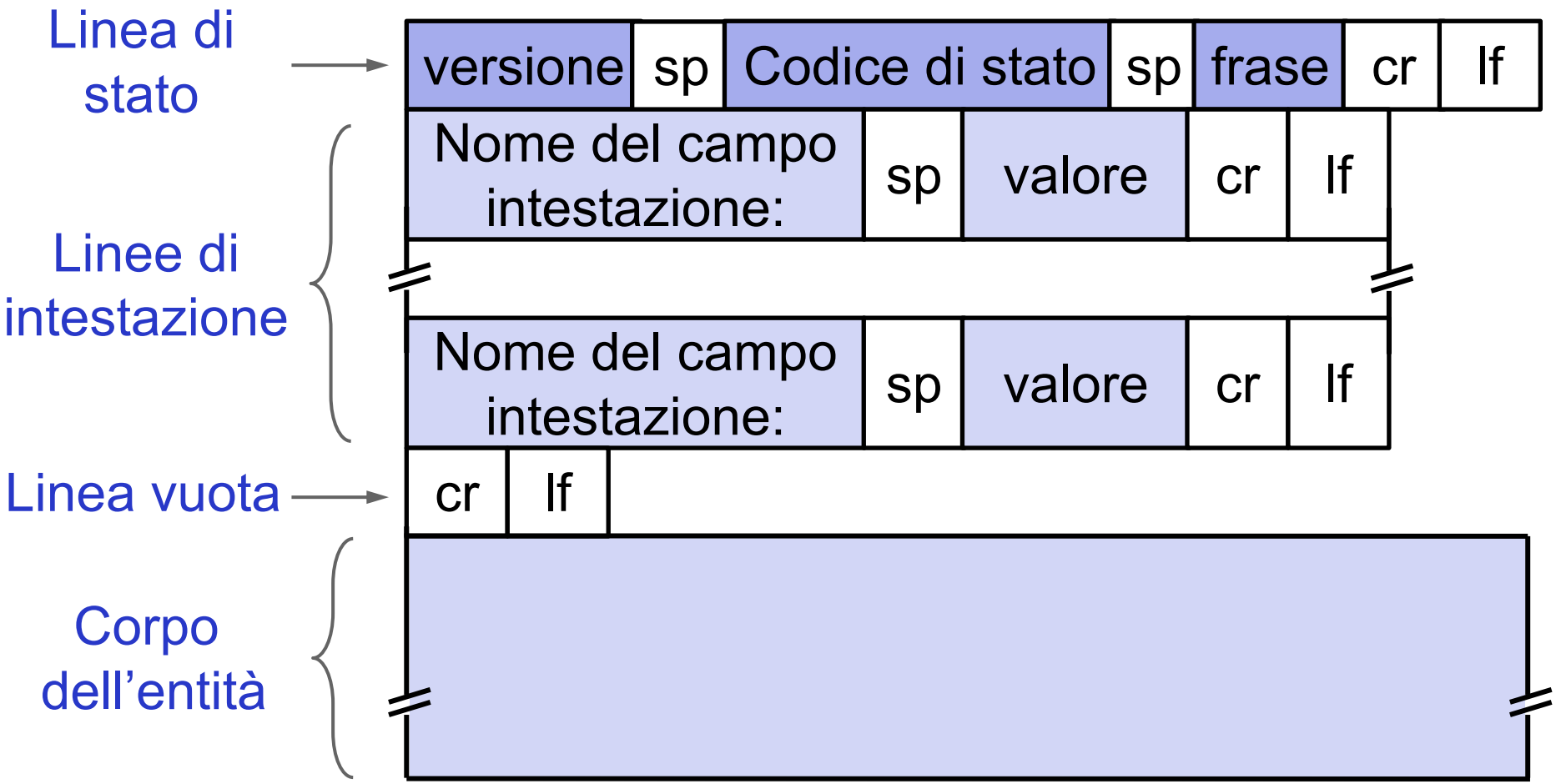
Metodi di richiesta

- Il metodo di richiesta, per la versione 1.1, può essere uno dei seguenti:
  - GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT

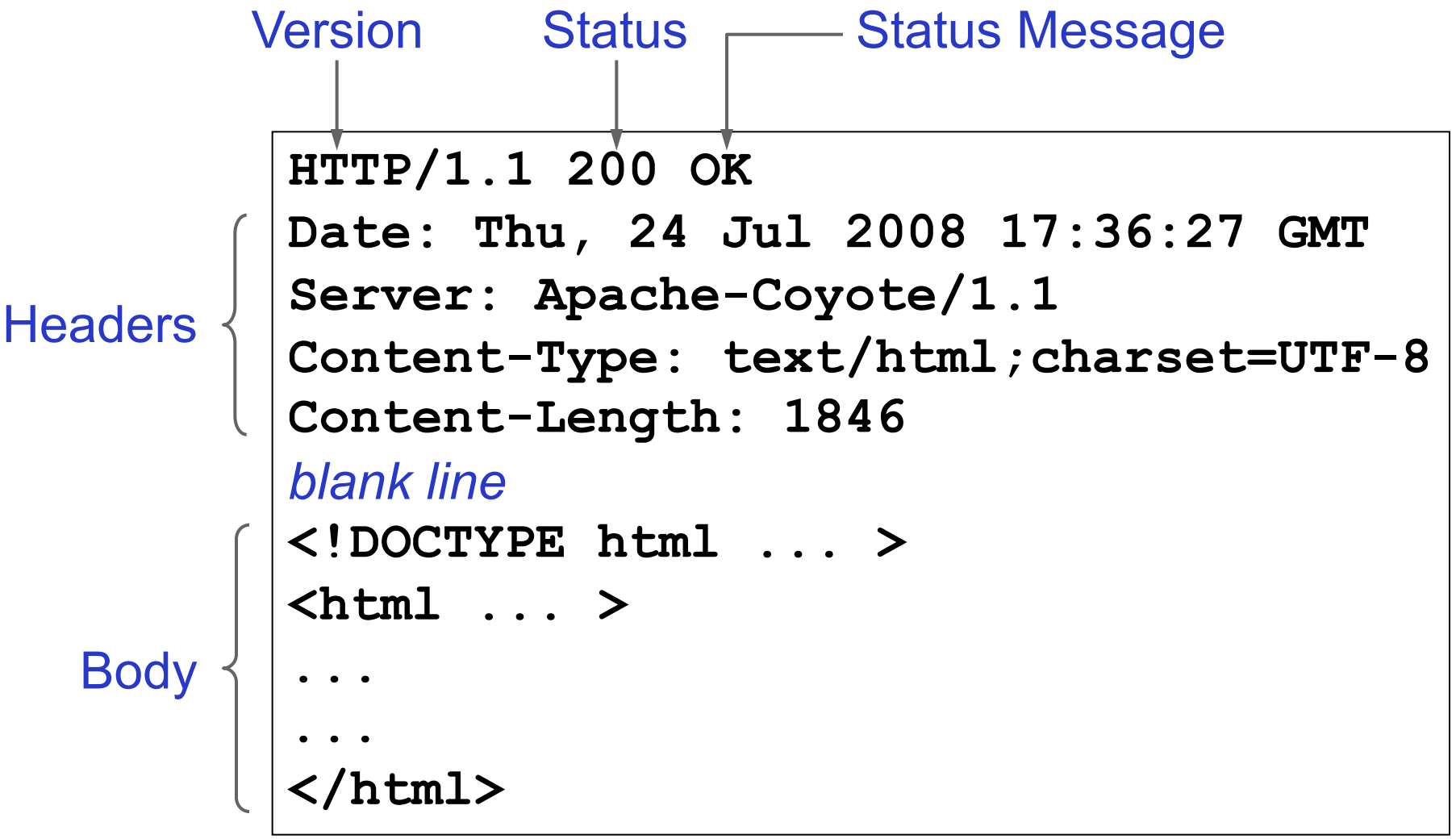
  Un metodo HTTP può considerarsi un comando proprio del protocollo HTTP che il client richiede al server.

- I più usati nelle web app sono **GET** e **POST**
- **GET** è usato per ottenere il contenuto della risorsa indicata come URI
- **POST** è usato per inviare informazioni al server (ad esempio: dati di un formulario)

Formato generale risposta HTTP



Esempio risposta HTTP



Header di risposta

- Gli header della risposta più comuni sono:
  - **Server**: Indica il tipo e la versione del server. Può essere visto come l'equivalente dell'header di richiesta User-Agent
  - **Content-Type**: Indica il tipo di contenuto restituito. Essi sono detti tipi **MIME** (Multimedia Internet Mail Extensions). Per esempio:
    - **text/html** Documento HTML
    - **application/json** Documento JSON
    - **application/xml** Documento XML
    - **image/jpeg** Immagine di formato JPEG
  - **Content-Length**: lunghezza in byte del body
  - Altri: **Content-Encoding**, **Content-Language**, **Last-Modified**

Content-type

- `Contet-type` descrive il *MIME type* del corpo del messaggio
- Un *MIME type* è un nome standardizzato che descrive il tipo di contenuto presente nel messaggio (text/html, image/gif, …)
- È usato dai *client* per capire come gestire il cotenuto ricevuto in una risposta HTTP
- Supporta parametri opzionali, per esempio *charset*:
  - `Content-type: text/html; charset=iso-8859-4`

Content-length

- L'header `Content-length` indica la dimensione in bytes del contenuto del corpo del messaggio
- La dimensione include qualsiasi tipo di encoding sia avvenuto sul contenuto (la dimensione di un testo compresso con gzip è quella compressa, non l'originale)
- Questo header è obbligatorio per messaggi con un corpo
- Serve a capire quando un messaggio è stato ricevuto per intero (o per capire quando un messaggio è arrivato troncato, se il *server* va in crash nel frattempo)

Entity body

- Il contenuto di una richesta/risposta HTTP inizia immediatamente dopo una linea vuota (CRLF) che marca la fine delle linee di intestazione (header fields)
- Il contenuto può essere qualsiasi cosa:
  - Testo
  - Binario
  - Documento
  - Immagine
  - Video
  - Compresso / non compresso
  - In Inglese / francese / giapponese (e quindi testo con caratteri “strani”)
  - ….

Status code della risposta

- 100–199: Informational Status Codes
- 200–299: Success Status Codes
  - **200 OK**
- 300–399: Redirection Status Codes
  - **301 Moved Permanently, 303 See Other, 304 Not Modified**
- 400–499: Client Error Status Codes
  - **400 Bad Request**
  - **401 Unauthorized**
  - **403 Forbidden**
  - **404 Not Found**
- 500–599: Server Error Status Codes
  - **500 Internal Server Error**

Unicode

- **Unicode** è un sistema di codifica che assegna un numero univoco ad ogni carattere usato per la scrittura di testi, in maniera **indipendente** dalla **lingua**, dalla **piattaforma informatica** e dal **programma utilizzato**.

- **UTF-8** (Unicode Transformation Format, 8 bit) è una codifica dei caratteri Unicode in sequenze di lunghezza variabile di byte. UTF-8 usa da 1 a 4 byte per rappresentare un carattere Unicode. Per esempio un solo byte è necessario per rappresentare i 128 caratteri dell'alfabeto **ASCII**.

- UTF-8 è diventato la codifica di caratteri **dominante per il World Wide Web**, contando più della metà di tutte le pagine web

- The W3C raccomanda UTF-8 come **encoding di default** nei loro standard (XML and HTML).

HTTPS

- **HyperText Transfer Protocol over Secure Socket Layer** (HTTPS) è il risultato dell'applicazione di un protocollo di crittografia asimmetrica al protocollo HTTP

- Viene utilizzato per garantire trasferimenti riservati di dati nel web, in modo da impedire intercettazioni dei contenuti che potrebbero essere effettuati tramite la tecnica di attacco del man in the middle.

- La sua porta di default è la 443

HTTP/2

- HTTP/2 è la nuova versione del protocollo di rete HTTP
- è basato su SPDY (protocollo di Google)
- la sua specifica è stata pubblicata in maggio 2015 (RFC 7540)
- Per ridurre la latenza usa degli header compressi e tecnologie *server push* così da ridurre il tempo di caricamento di una pagina
- SPDY ha dimostrato evidenti miglioramenti rispetto a HTTP, soprattutto nella velocità di caricamento delle pagine (dal 11.81% al 47.7%)
- Alcuni browser lo implementano già: http://caniuse.com/#search=http%2F2
- E anche alcuni web/application servers: https://github.com/http2/http2-spec/wiki/Implementations

HTTP/2

- Mantenere la compatibilità di alto livello con HTTP 1.1 (per esempio con i metodi, codici di stato, e URI, e la maggior parte campi di intestazione)
- Ridurre la latenza per migliorare la velocità di caricamento delle pagine nei browser web considerando:
  - La compressione dei dati degli HTTP headers
  - Tecnologie server push
  - Risolvere il problema head-of-line di blocco in HTTP 1

  (http://stackoverflow.com/questions/10480122/difference-between-http-pipeling-and-http-multiplexing-with-spdy)
  - Caricamento degli elementi in parallelo di una pagina su una singola connessione TCP
- Sostenere casi d'uso comuni di HTTP, come browser desktop, browser per cellulari, API web, server web a varie scale, server proxy, …

Fonti

- HTTP: The Definitive Guide, David Gourley and Brian Totty, O'Reilly, 2002
- Internet e reti di calcolatori, James Kurose and Keith Ross, McGraw-Hill, 2013
- http://www.w3.org/Protocols/rfc2616/rfc2616.html
- http://it.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- http://it.wikipedia.org/wiki/Uniform_Resource_Identifier
- http://it.wikipedia.org/wiki/Uniform_Resource_Locator
- http://www.unicode.org/
- http://en.wikipedia.org/wiki/Unicode_and_HTML
- https://it.wikipedia.org/wiki/HTTPS
- https://it.wikipedia.org/wiki/HTTP/2