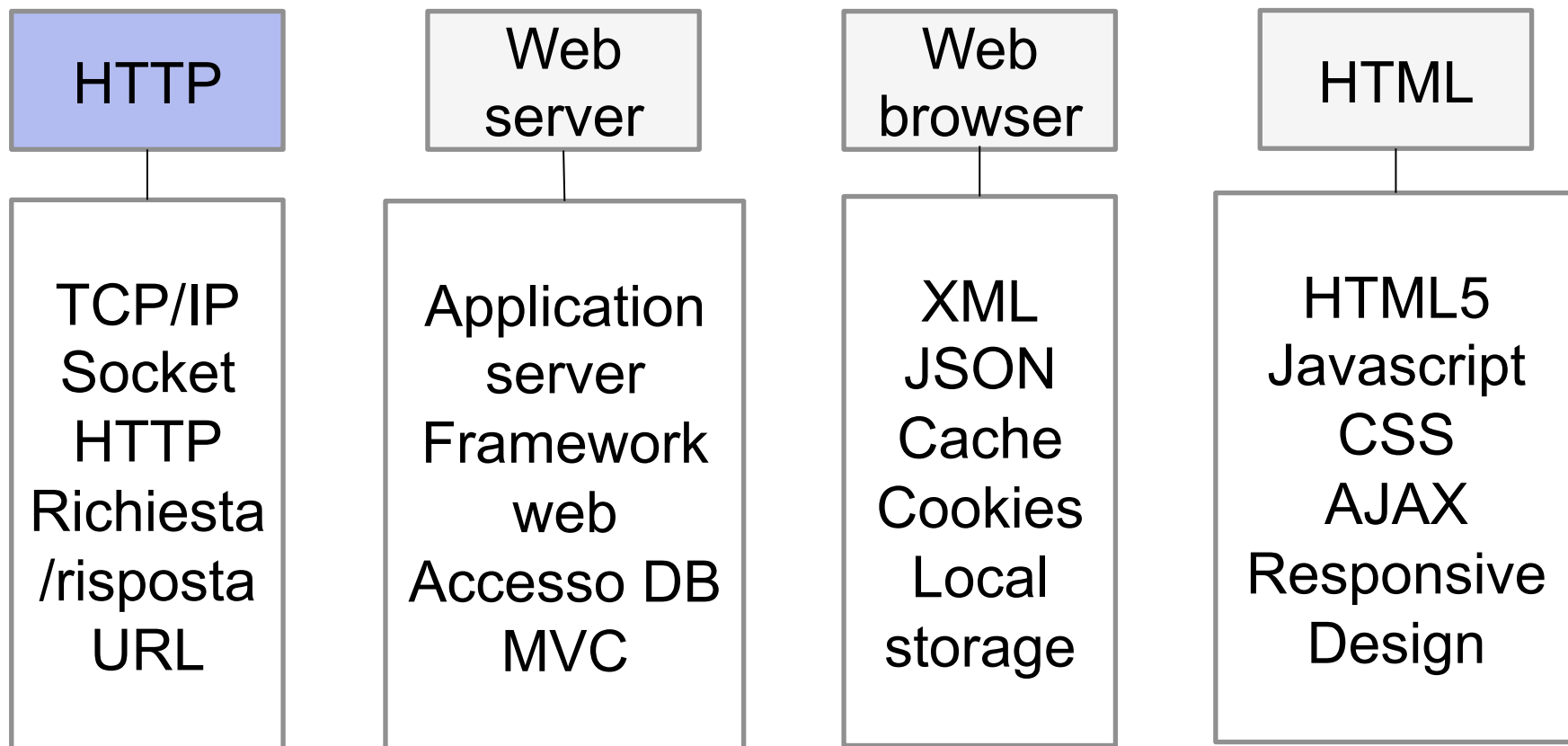


**SUPSI**

# Protocollo HTTP

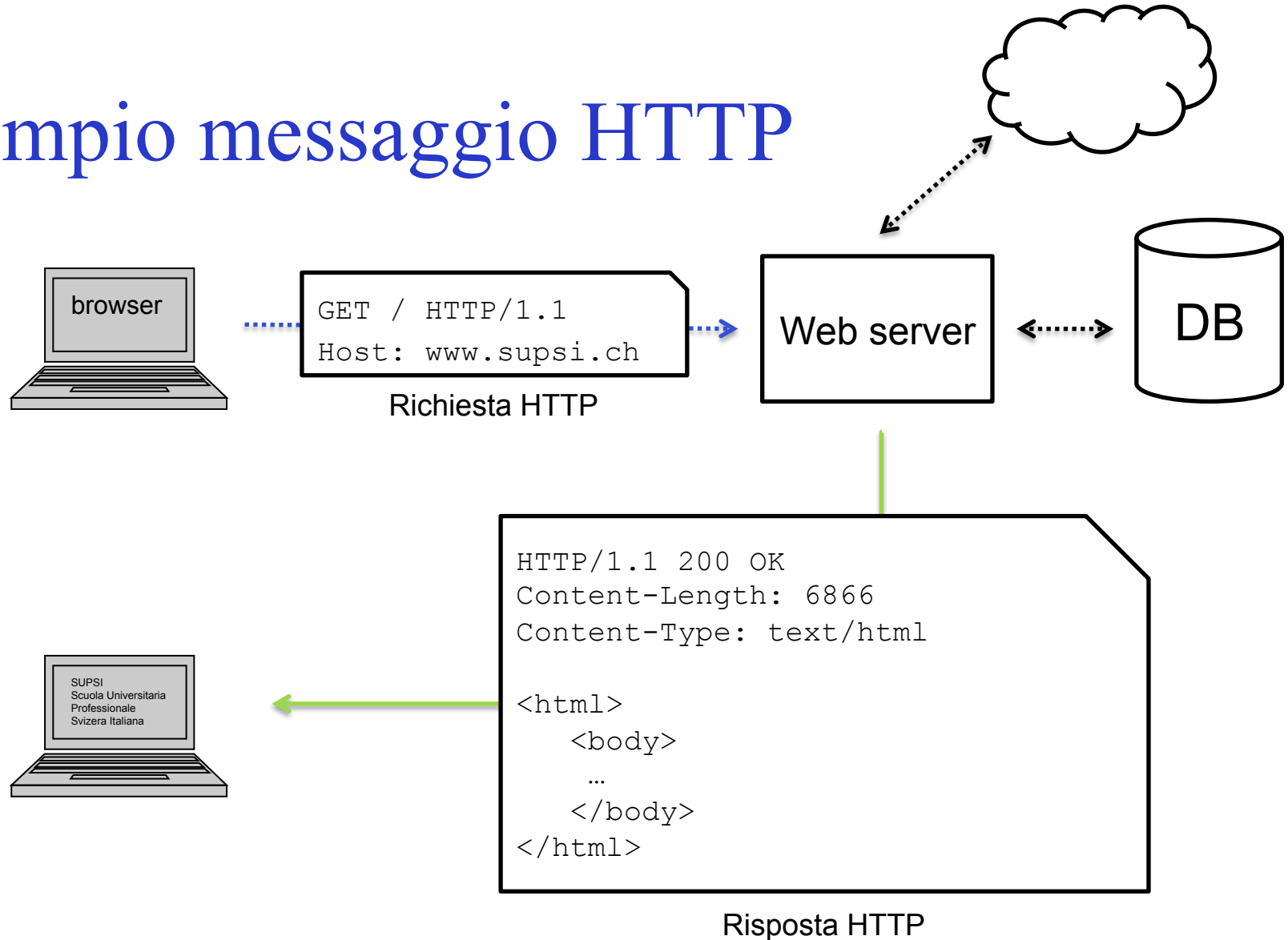
# Componenti di una web app



# 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.

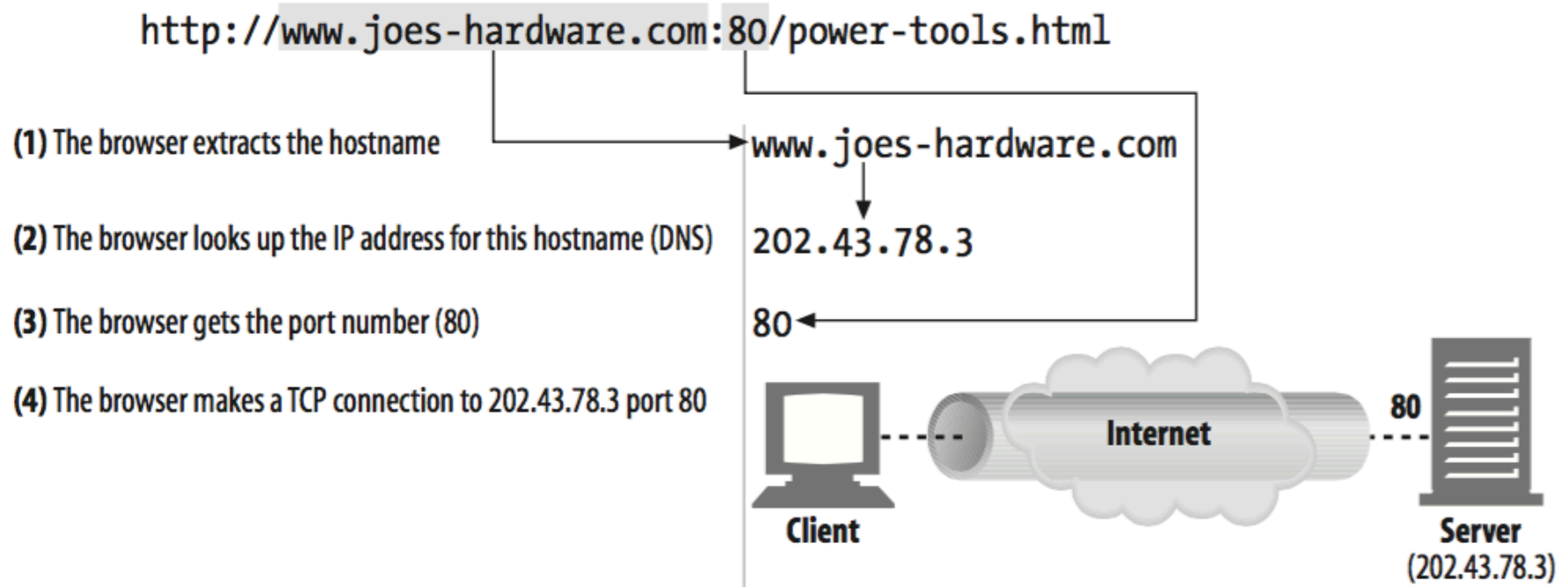
# Esempio messaggio HTTP



# Esempio messaggio HTTP

- Il browser del client invia al server una richiesta HTTP, specificando l'indirizzo (URL) della risorsa su una determinata 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

# Esempio messaggio HTTP

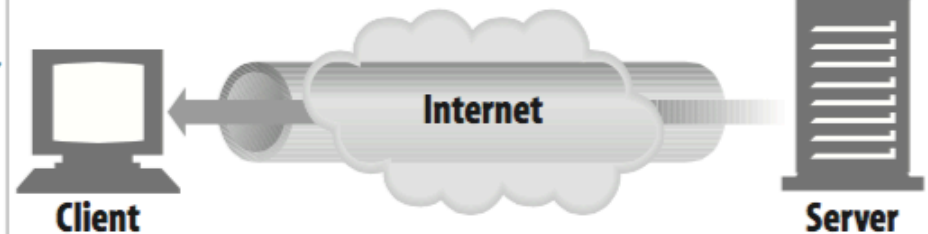


# Esempio messaggio HTTP

(5) The browser sends an HTTP GET request message to the server



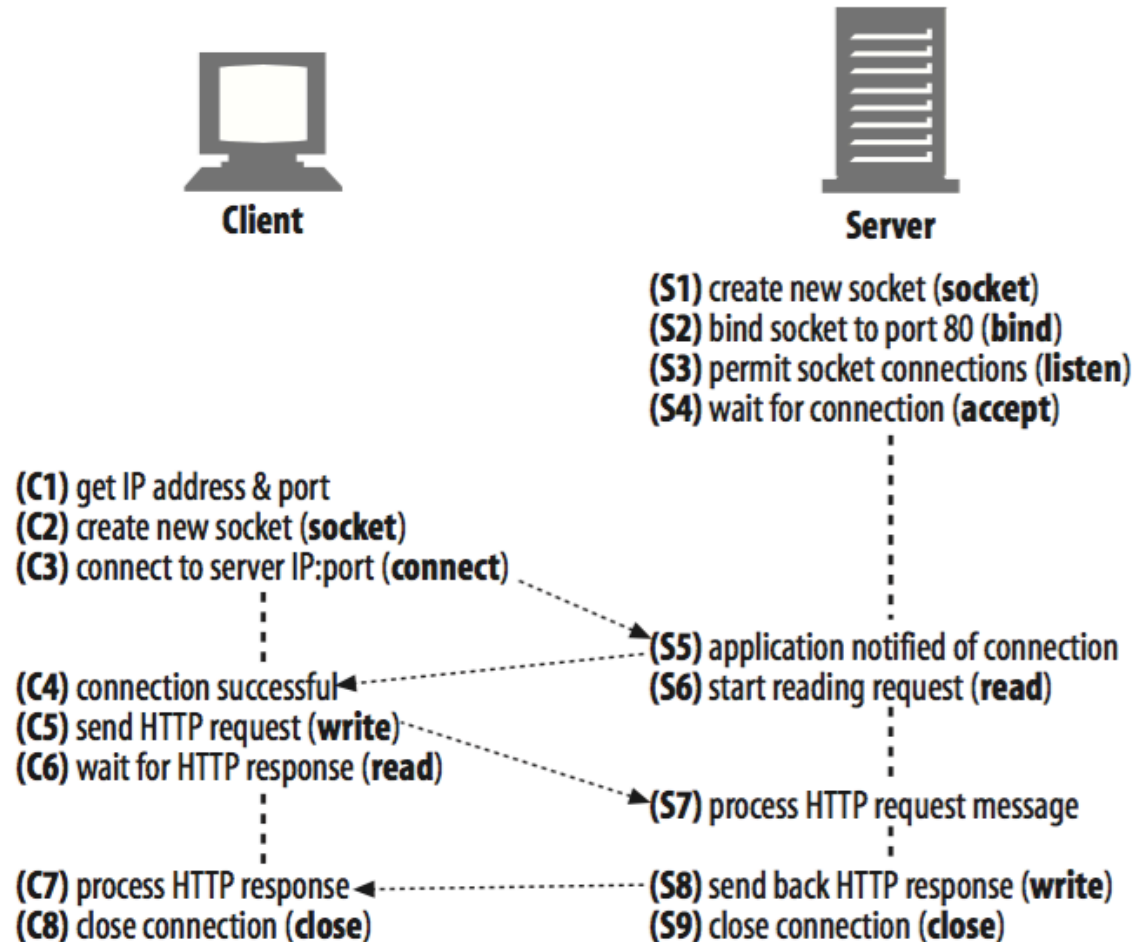
(6) The browser reads the HTTP response message from the server



(7) The browser closes the connection



# Esempio comunicazione TCP

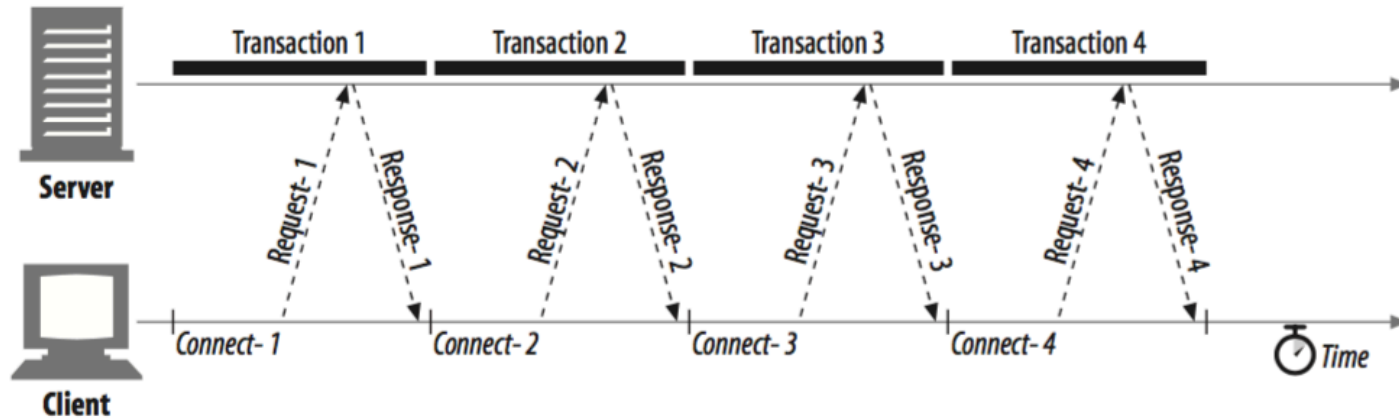


Fonte: **HTTP: The Definitive Guide**

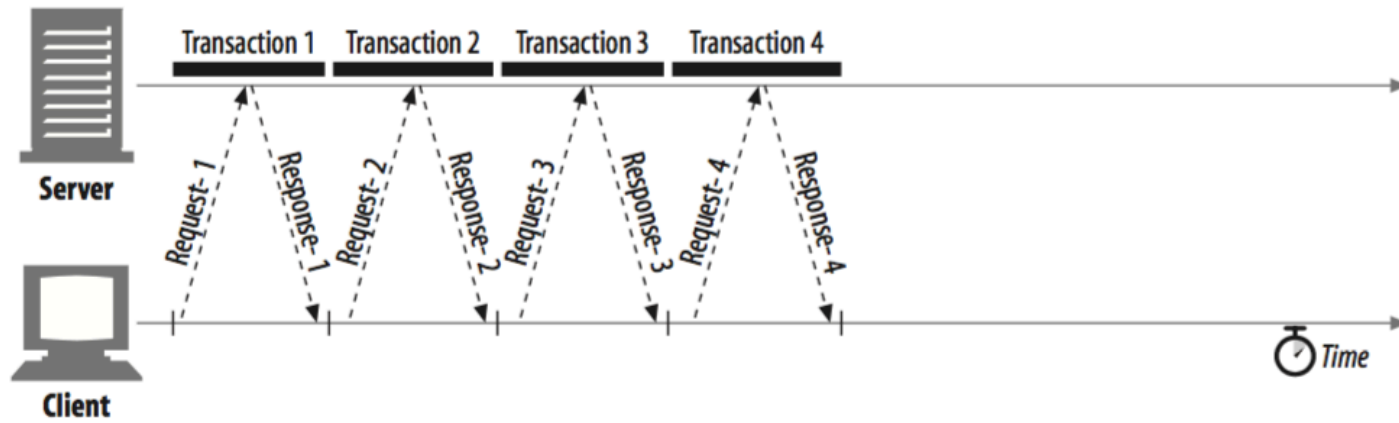


# Da HTTP 1 a HTTP 1.1

(a) Serial connections



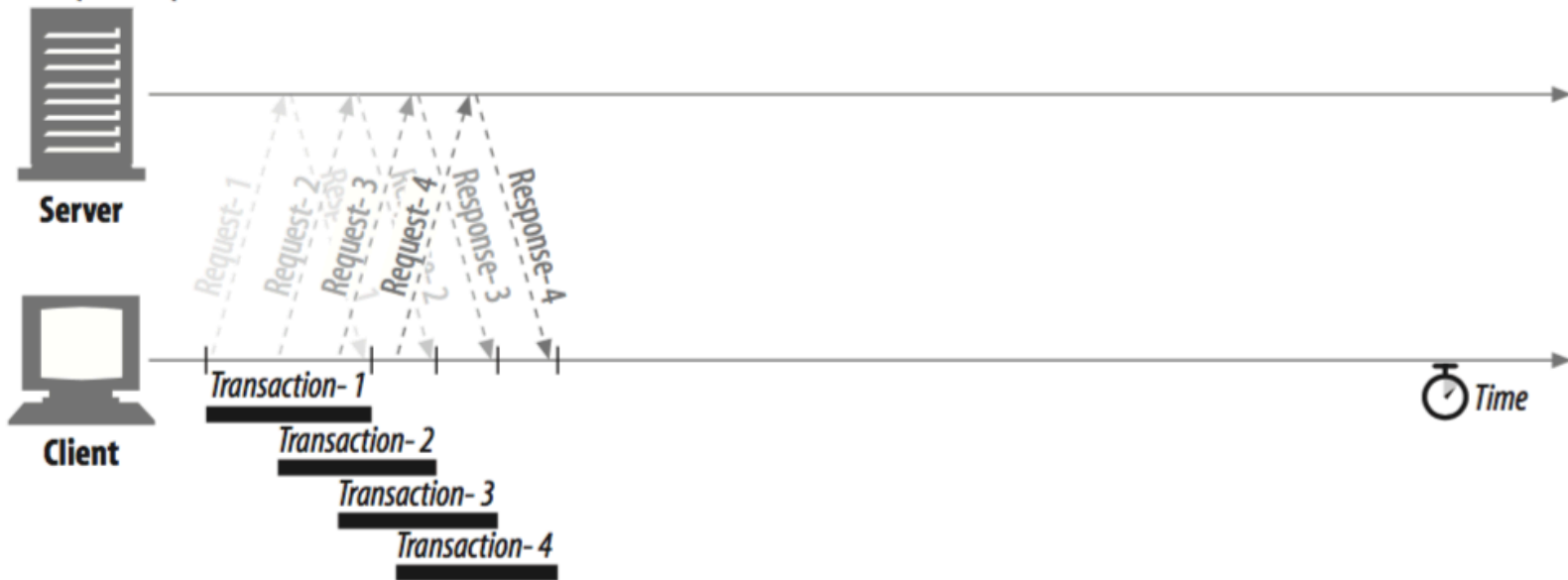
(b) Persistent connection



Fonte: HTTP: The Definitive Guide

# Fino a pipelined connections

(c) Pipelined, persistent connection



Fonte: **HTTP: The Definitive Guide**

# URI (Uniform Resource Identifier)

- Per poter individuare ed accedere a documenti (risorse) su WEB è necessario un meccanismo per poterli identificare e localizzare
- Un **URI** può essere classificato come **URL** o come **URN**
- Un **URL** (Uniform Resource Locator) indica la collocazione reale di un oggetto accessibile mediante uno dei protocolli attualmente in uso su Internet.
- Un **URN** (Uniform Resource Name) è un URI che identifica una risorsa mediante un "nome" in un particolare dominio di nomi ("namespace"). Per esempio: **urn:isbn:0-395-36341-1**

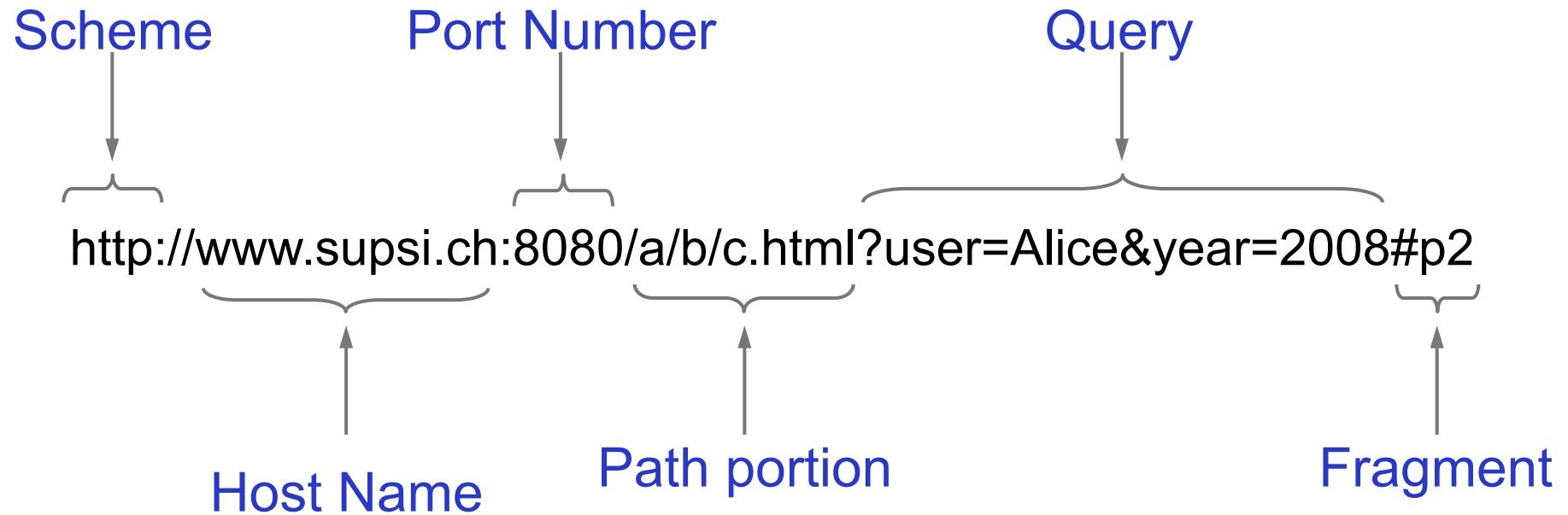
# URL (Uniform Resource Locator)

- 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

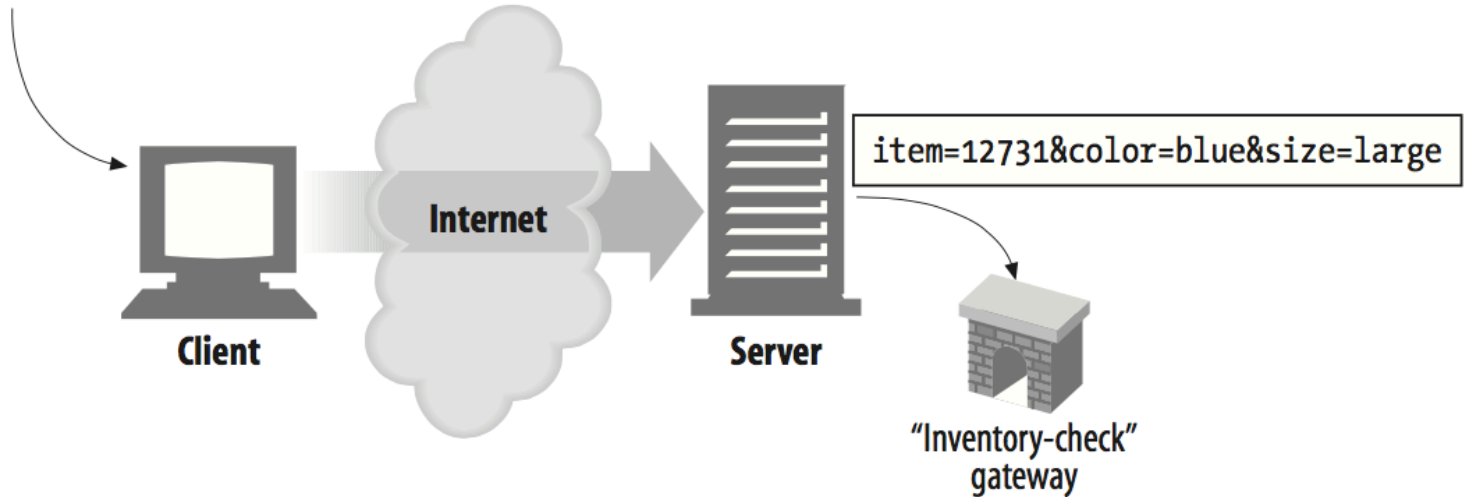


# Esempi di URL

- <http://www.supsi.ch>
- <http://localhost:8080/index.html>
- <https://www.google.ch/search?q=supsi&hl=it>
- [http://it.wikipedia.org/wiki/  
Scuola\\_universitaria\\_professionale\\_della\\_Svizzera\\_italiana#Bachelor](http://it.wikipedia.org/wiki/Scuola_universitaria_professionale_della_Svizzera_italiana#Bachelor)

# Query string

`http://www.joes-hardware.com/inventory-check.cgi?item=12731&color=blue&size=large`



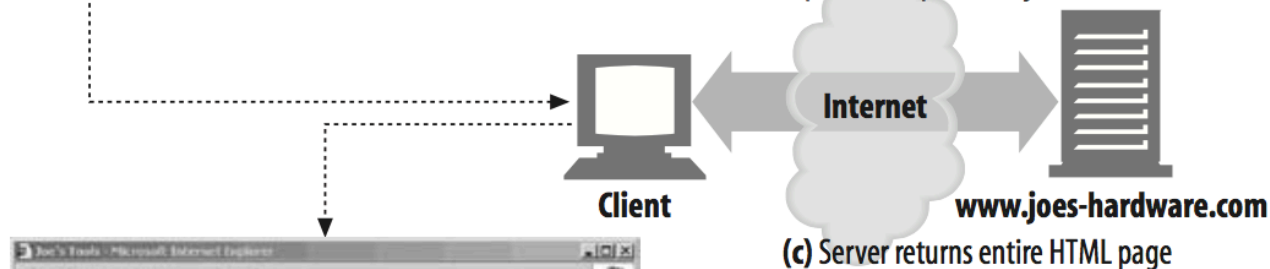
# Fragment

`http://www.joes-hardware.com/tools.html#drills`

(a) User selects link to "`http://www.joes-hardware.com/tools.html#drills`"

*(Fragment is NOT sent to the server)*

(b) Browser makes request to `http://www.joes-hardware.com/tools.html`



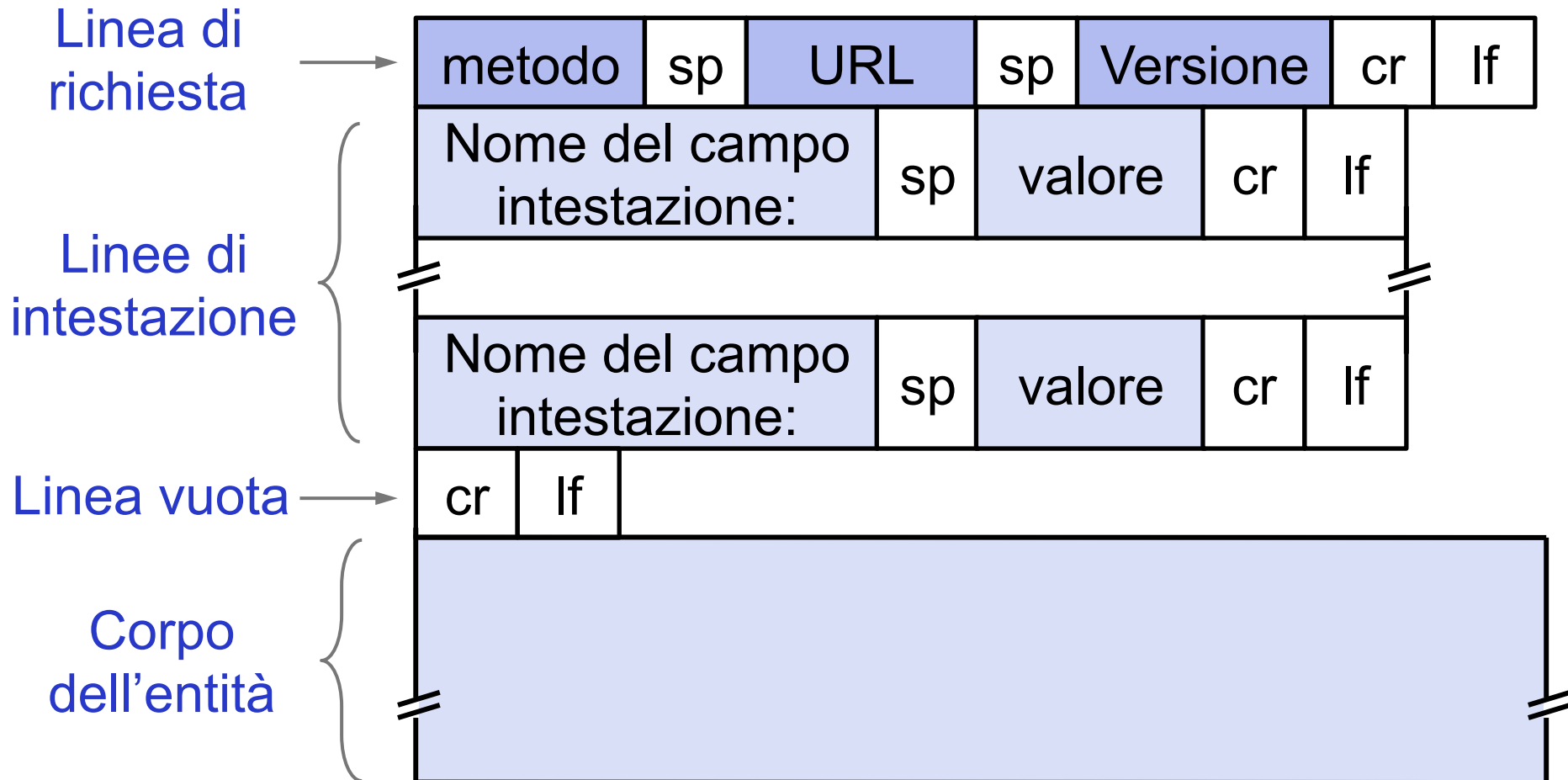
(c) Server returns entire HTML page

*Browser scrolls down to start at named "drills" fragment*

(d) Browser displays HTML page starting with named "drills" fragment



# 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**

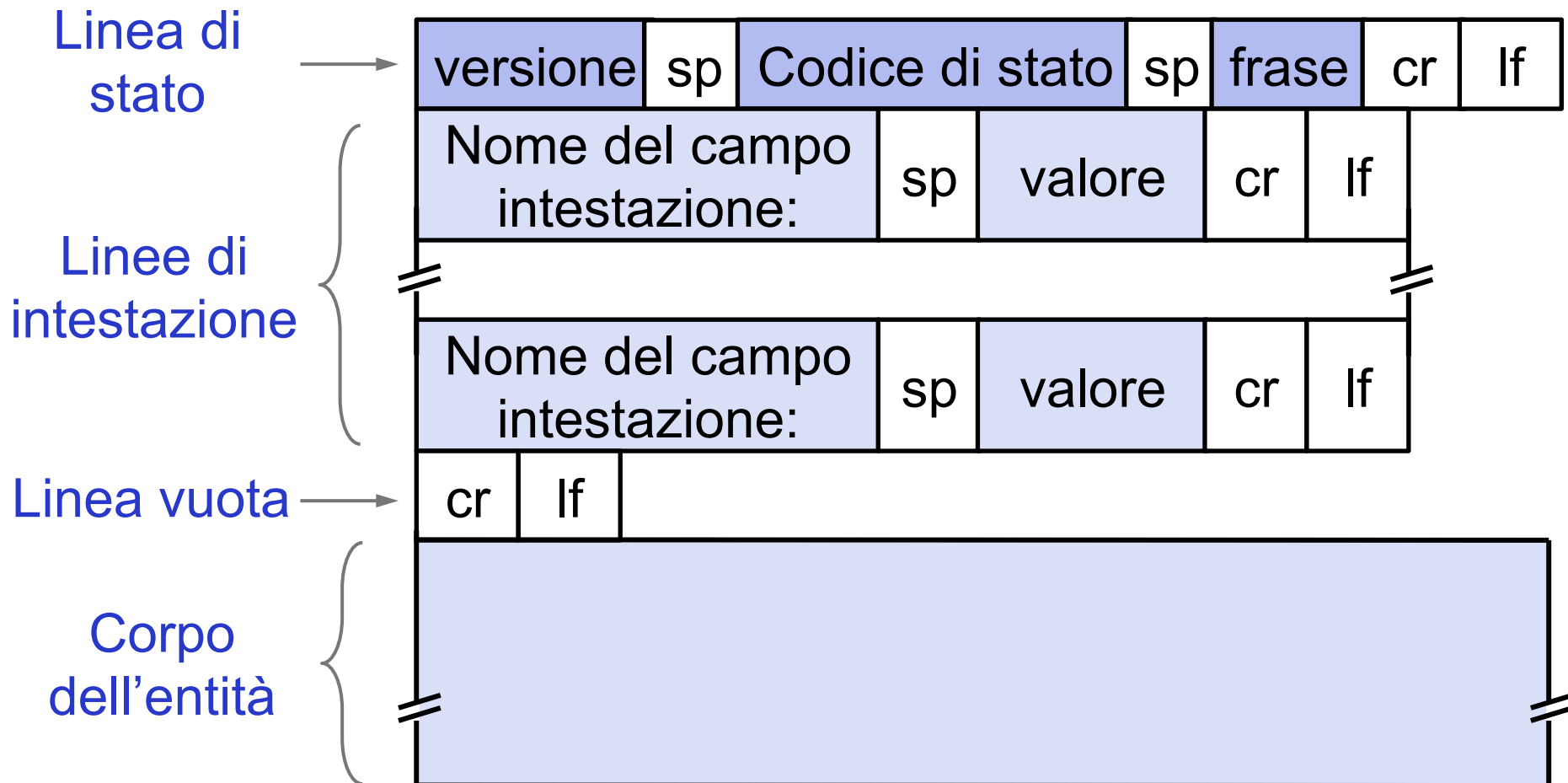
# Metodi di richiesta

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

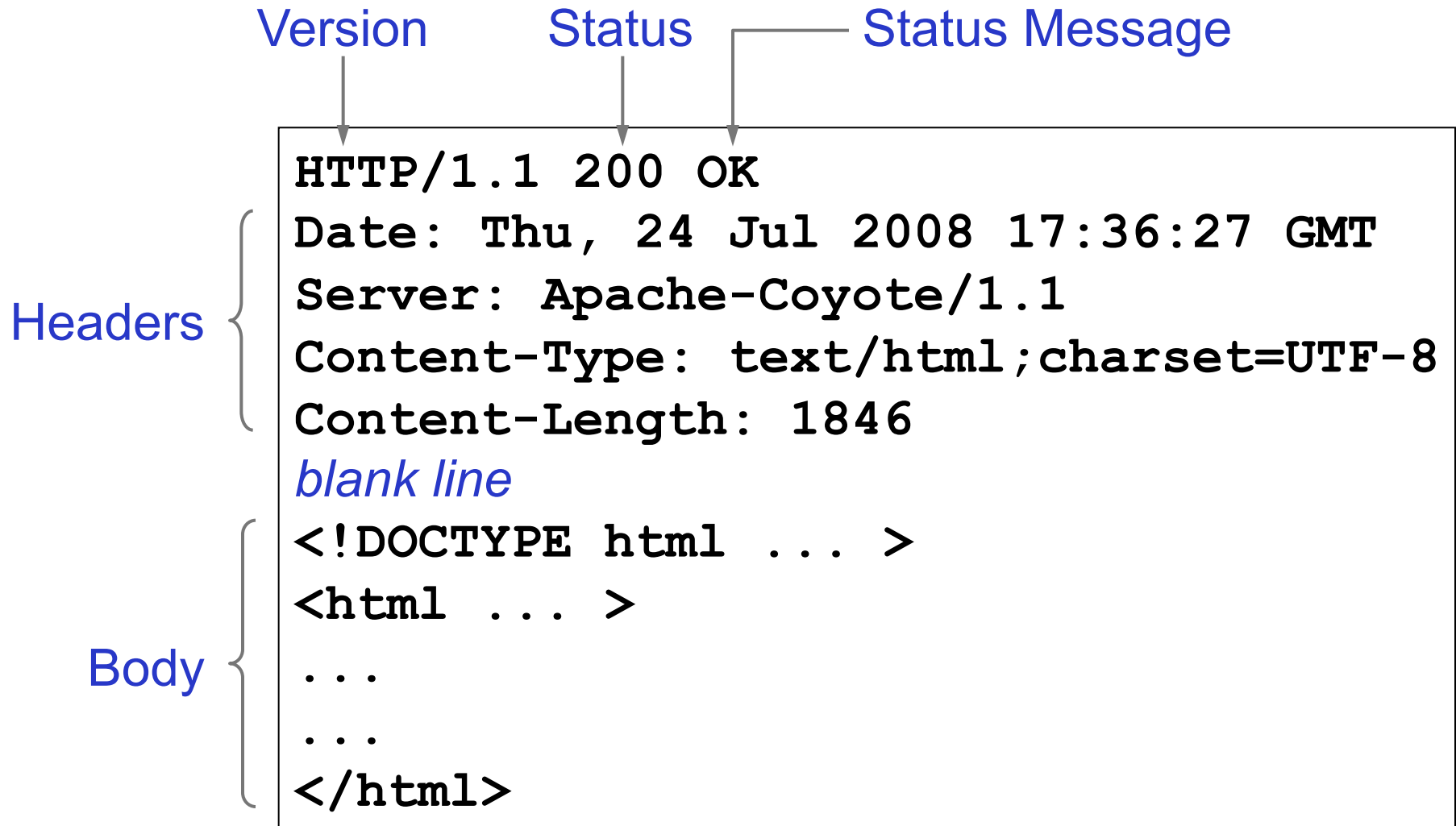
Un metodo HTTP può considerarsi un comando proprio del protocollo HTTP che il client richiede al server.

- 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

- `Content-type` describe il *MIME type* del corpo del messaggio
- Un *MIME type* è un nome standardizzato che describe il tipo di contenuto presente nel messaggio (text/html, image/gif, ...)
- È usato dai *client* per capire come gestire il contenuto 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)

# Content-encoding

- Alcune volte le risposte dai server potrebbero arrivare codificate
- Per esempio per:
  - Comprimere un file troppo grosso su una connessione lenta
  - Cifrare un documento prima di inviarlo
- Il `Content-type` è comunque presente e indica qual'era il formato originale del corpo del messaggio

# Content-encoding

## Original content

Content-type: text/html  
Content-length: 12480



## Content-encoded content

Content-type: text/html  
Content-length: 3907  
Content-encoding: gzip

01001011  
11000101

## Original content

Content-type: text/html  
Content-length: 12480



Fonte: HTTP: The Definitive Guide

# Accept-encoding

- Cosa succede però se un server codifica una risposta, ma il *client* non è capace a decodificarla?
- Per risolvere il problema il *client* può inviare nella richiesta una lista di codifiche che può accettare, usando l'header `Accept-encoding`
- Per esempio:
  - `Accept-encoding: gzip`

# Entity body

- Il contenuto di una richiesta/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”)
  - ....

# Entity body

## (a) Text/plain entity in HTTP response message

|   |  |  |
|---|--|--|
| HTTP/1.0 200 OK<br>Content-type: text/plain<br>Content-length: 18 | 00: 4854 5450 2f31 2e30 2032 3030 204f 4b0d<br>16: 0a43 6f6e 7465 6e74 2d74 7970 653a 2074<br>32: 6578 742f 706c 6169 6e0d 0a43 6f6e 7465<br>48: 6e74 2d6c 656e 6774 683a 2031 380d 0a0d<br>64: 0a48 6921 2049 276d 2061 206d 6573 7361<br>80: 6765 210a | HTTP/1.0 200 OK.<br>.Content-type: t<br>ext/plain..Conte<br>nt-length: 18...<br>[Hi! I'm a messa<br>ge!] |
|---|--|--|

final LF (0x0A= <LF>)      start-of-content (0x48= "H")

## (b) Image/gif entity in HTTP response message

|   |   |  |
|---|---|--|
| HTTP/1.0 200 OK<br>Content-Type: image/gif<br>Content-Length: 34867 | 00: 4854 5450 2f31 2e30 2032 3030 204f 4b0d<br>16: 0a43 6f6e 7465 6e74 2d74 7970 653a 2069<br>32: 6d61 6765 2f67 6966 0d0a 436f 6e74 656e<br>48: 742d 6c65 6e67 7468 3a20 3334 3836 370d<br>64: 0a0d 0a47 4946 3837 6127 0206 02f7 0000<br>80: 0402 0404 8204 8482 04fc fe04 8402 0404<br>96: 42a4 a4ca f484 8284 dce2 dc9c a29c 444a | HTTP/1.0 200 OK.<br>.Content-type: i<br>mage/gif..Conten<br>t-length: 34867.<br>...GIF87a'.....<br>.....<br>B.....DJ |
|---|---|--|

final LF (0x0A= <LF>)      start-of-content ("GIF87a")      Width (0x0227= 551)      Height (0x0206= 518)

Fonte: **HTTP: The Definitive Guide**

# 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**

# 304 Not Modified



Il browser visualizza una copia **locale**, visto che l'originale non è cambiato

Fonte: **HTTP: The Definitive Guide**



# 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-pipelining-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/Hypertext_Transfer_Protocol)
- [http://it.wikipedia.org/wiki/Uniform\\_Resource\\_Identifier](http://it.wikipedia.org/wiki/Uniform_Resource_Identifier)
- [http://it.wikipedia.org/wiki/Uniform\\_Resource\\_Locator](http://it.wikipedia.org/wiki/Uniform_Resource_Locator)
- <http://www.unicode.org/>
- [http://en.wikipedia.org/wiki/Unicode\\_and\\_HTML](http://en.wikipedia.org/wiki/Unicode_and_HTML)
- <https://it.wikipedia.org/wiki/HTTPS>
- <https://it.wikipedia.org/wiki/HTTP/2>