Appunti su HTTP (HyperText Transfer Protocol)

Introduzione

- **HTTP** è un protocollo *client-server*, *generico* e *stateless*.
  - **Client-server:** Il client inizia la connessione, richiede servizi. Il server accetta, identifica (opzionalmente) e risponde, poi chiude la connessione.
  - **Generico:** Indipendente dal formato dei dati (HTML, binari, eseguibili, ecc.).
  - **Stateless:** Il server non mantiene informazioni tra le connessioni. Il client deve ricreare il contesto ad ogni richiesta.

Concetto Chiave: Risorse HTTP

- HTTP scambia **risorse** identificate da **URI**.
- Separa le **risorse** dalla loro **rappresentazione**.
- Permette la **negoziazione del formato** (stessa risorsa, formati diversi).
- Implementa politiche di **caching** per migliorare le performance (copie su proxy, gateway, ecc.).

Connessione HTTP

- Serie di richieste e risposte.
- Connessioni **persistenti** con:
  - **Pipelining:** Più richieste senza attendere risposte. Le risposte arrivano nello stesso ordine delle richieste.
  - **Multiplexing:** Richieste e risposte multiple, anche in ordine diverso, ricostruite dal client.
- **HTTP/2** introduce:
  - **Operazioni PUSH:** Il server anticipa le richieste del client.
  - **Compressione degli header:** Dati compressi e inviati in parallelo.

Richieste e Risposte HTTP

Struttura generale:

```
Headers
Body
Request/Status Line (all'inizio)
Client/Proxy/Gateway <---> Origin Server/Proxy/Gateway
```

La Richiesta HTTP

Componenti:

1. **Method:** Azione richiesta (GET, POST, PUT, DELETE, ecc.).
2. **URI:** Identificativo della risorsa.
3. **Version:** Versione di HTTP.
4. **Headers:**
   - Generali (es: Connection, Date).
   - Di richiesta (es: User-Agent, Referer, Host).
   - Di entità (es: Content-Type, Content-Length).
5. **Body:** Messaggio MIME (opzionale).

Esempio di richiesta:

```
GET /beta.html HTTP/1.1
Referer: http://www.alpha.com/alpha.html
Connection: Keep-Alive
User-Agent: Mozilla/4.61 (Macintosh; I; PPC)
Host: www.alpha.com:80
Accept: image/gif, image/jpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

Metodi HTTP (Verbi)

- Indicano l'azione sulla risorsa (o sulla sua rappresentazione).
- Importanti per interoperabilità e caching.
- Principali: GET, HEAD, POST, PUT, DELETE, OPTIONS, PATCH.

Esempi di GET e POST

- **GET**: Richiede una risorsa. Attivato da click su link o inserimento URL.
  - Esempio: `GET /courses/tw.html`
- **POST**: Invia dati al server relativi alla risorsa.
  - Esempio:

    ```
    POST /courses/1678
    {
      "titolo":"Tecnologie Web",
      "descrizione":"Il corso..bla..bla.."
    }
    ```

La Risposta HTTP

Componenti:

1. **Status code:** Indica successo o fallimento (e il tipo).
2. **Version:** Versione di HTTP.
3. **Headers:**
   - Generali.
   - Di risposta (es: Server, WWW-Authenticate).
   - Di entità.
4. **Body:** Messaggio MIME (opzionale).

Esempio di risposta:

```
HTTP/1.1 200 OK
Date: Fri, 26 Nov 2007 11:46:53 GMT
Server: Apache/1.3.3 (Unix)
Last-Modified: Mon, 12 Jul 2007 12:55:37 GMT
Accept-Ranges: bytes
Content-Length: 3357
Content-Type: text/html
```

```
<HTML> …. </HTML>
```

Status Code

- Codice a 3 cifre.
- Prima cifra: classe della risposta.
  - **1xx:** Informational (risposta temporanea).
  - **2xx:** Successful (richiesta accettata).
  - **3xx:** Redirection (azioni aggiuntive necessarie).
  - **4xx:** Client error (errore del client).
  - **5xx:** Server error (problema del server).

Esempi di Status Code:

- `100 Continue`
- `200 OK`
- `201 Created`
- `301 Moved Permanently`
- `400 Bad Request`
- `401 Unauthorized`
- `403 Forbidden`
- `404 Not Found`
- `500 Internal Server Error`

Utilità dello Status Code:

- API più chiare e semplici.
- Il client capisce l'esito senza leggere il body.
- Migliora caching, redirezione, uniformità e interoperabilità.

Gli Header

- Righe di testo (RFC822) con informazioni aggiuntive.
- Presenti in richieste e risposte.

Tipo Header	Descrizione	Richiesta	Risposta
Generali	Informazioni sulla trasmissione	Sì	Sì
Di richiesta	Informazioni sulla richiesta	Sì	No
Di risposta	Informazioni sulla risposta generata	No	Sì
Di entità	Informazioni sulla risorsa e i dati trasmessi	Sì	Sì

- **Header Generali (Esempi):**
  - `Date`
  - `Transfer-Encoding`
  - `Cache-Control`
  - `Connection`
- **Header dell'Entità (Esempi):**
  - `Content-Type` (obbligatorio con body)
  - `Content-Length` (obbligatorio, specialmente con connessioni persistenti)

    - `Content-Encoding`, `Content-Language`, `Content-Location`, ecc.
- **Header della Richiesta (Esempi):**
    - `User-Agent`
    - `Referer`
    - `Host`
    - (Altri per cache e autenticazione)
- **Header della Risposta (Esempi):**
    - `Server`
    - `WWW-Authenticate`

Importanza del `Content-Type`:

- Indica al client come processare la risorsa.
- `Content-Type` e `Content-Length` sono obbligatori se la risposta ha un body.

Metodi HTTP (di nuovo)

- **Sicurezza:** Un metodo è sicuro se non cambia lo stato del server (eccetto i log). Può essere eseguito da nodi intermedi senza problemi.
- **Idempotenza:** Più richieste identiche hanno lo stesso effetto di una sola (eccetto i log). Può essere rieseguito senza problemi.

Tabella riassuntiva dei metodi principali:

Metodo	Descrizione	Sicuro	Idempotente
`GET`	Richiede una risorsa.	Sì	Sì
`HEAD`	Come `GET`, ma restituisce solo gli header.	Sì	Sì
`POST`	Invia dati al server. Può creare risorse.	No	No
`PUT`	Crea o sostituisce una risorsa.	No	Sì
`DELETE`	Rimuove una risorsa.	No	Sì
`PATCH`	Aggiorna parzialmente una risorsa.	No	No
`OPTIONS`	Verifica opzioni, requisiti e servizi del server. Usato per CORS (Cross-Origin Resource Sharing).	Sì	Sì

Conclusioni

- Viste le caratteristiche principali di HTTP.
- Argomenti futuri:
    - Cookies
    - Cross-site Origin
    - Caching
    - Authentication
    - HTTP/2 e HTTP/3