

HTTP

Ing. Cesare Monti
(rivisto da Andrea Omicini)

Organizzazione della conoscenza

- Che succede se vogliamo rappresentare a pieno il pensiero umano?
- teoria dell'ipertesto
 - slegare la rappresentazione dal supporto fisico
 - catturare i nessi e non la sola informazione
 - focalizzarsi sull'informazione come oggetto

Organizzazione della conoscenza

- tappe storiche
 - 1588 - "Le diverse et artificiosae macchine del Capitano Agostino Ramelli"
 - 1945 - Vannevar Bush - consigliere di Roosevelt
 - As we may think - Atlantic Monthly
 - Memex - sistema basato sui microfilm
 - 1965 - Theodor Holm Nelson
 - conio il termine ipertesto
 - Xanadu - dal luogo magico di "Kubla Khan" di S. Taylor

Organizzazione della conoscenza

- 1990 - Tim Berners-Lee
 - World Wide Web
 - HTML
 - URLs
 - HTTP
- 1991/92
 - NCSA - Mosaic - primo browser

HTTP

- insieme di RPC (Remote Procedure Calls) basate su TCP/IP
- repository di risorse identificate da URL
- ogni risorsa è (può essere) un ipertesto
- ad ogni richiesta corrisponde una nuova connessione

Il protocollo HTTP

- Si basa su due fasi
 - Richiesta (HTTP:Request)
 - Risposta (HTTP:Response)
- Ognuna di queste fasi ha un suo possibile protocollo
 - Request: {line, header, body}
 - Response: {header line, headers fields, body}

HTTP:Request

- Request line (nome del comando da invocare, es: GET)
- Request header field (informazioni aggiuntive, es: parametri di RPC)
- Entity body (riservato al passaggio di informazioni "bulk" al server)

HTTP:Request

```
- Sintassi
<method><resource identifier><http version> <clrf> } Request line
[<Header> : <value>] <clrf> } Request
: : } Header
: : } Fields
[<Header> : <value>] <clrf>
blank line <clrf>
[Entity body] } Entity Body

- Esempio
GET /path/to/file.html HTTP/1.0 } Request line
Accept: text/html } Request
Accept: audio/x } Header
Accept: image/gif } Fields
User-agent: MacWeb }
```

HTTP: Request

- di Method ne esistono diversi !
 - **get**
 - richiesta semplice
 - **head**
 - = get ma restituisce solo le intestazioni
 - **post**
 - chiede di accettare i dati in stream
 - **put**
 - chiede di memorizzare i dati sulla risorsa URI specificata
 - **delete**
 - chiede di cancellare la risorsa URI specificata
 - **options**
 - chiede quali methods siano supportati dal server web
 - **trace**
 - chiede di mostrare la request http e le sue intestazioni
 - **connect**
 - ...mai implementato per l'utilizzo di proxy per operazioni di tunneling

HTTP:Response

- Response Header Line (protocollo e numero di errore)
- Response Header Field (informazioni aggiuntive, contenuto, lunghezza ecc...)
- Entity body (il corpo della pagina richiesta)

HTTP:Response

```
- Sintassi
<HTTP Version> <result code> [<explanation>] <clrf> } Response line
[<Header> : <value>] <clrf> } Response
: : } Header
: : } Fields
[<Header> : <value>] <clrf>
blank line <clrf>
[Entity body] } Entity Body
- Esempio
HTTP/1.0 200 OK } Response line
SERVER: NCSA/1.3 } Response
Mime_Version: 1.0 } Header
Content_type: text/html } Fields
Content_length: 2000
<HTML>
:
:
</HTML> } Entity Body
```

HTTP: limiti

- tra ogni richiesta non c'è interazione né dipendenza
- completa perdita di stato ad ogni connessione
 - protocollo stateless
 - questo deriva dalla sua progettazione
- ogni RPC è slegata dalle altre
 - non c'è modo di mantenere la storia dell'interazione
 - decade il legame spazio-tempo

HTTP verso i sistemi distribuiti

- Per ovviare al problema negli anni sono stati sviluppati differenti modelli di interazione
 - spostare capacità di calcolo sul server
 - sistemi a mainframe
 - spostare capacità di calcolo sul client
 - reti di PC
 - modelli ibridi
 - architetture client-server