Cosa succede quando si clicca un un link

I protocolli di comunicazione

Autori
Silvio Peroni – silvio.peroni@unibo.it
Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia

Aldo Gangemi – aldo.gangemi@unibo.it
Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia

Sommario

In questo capitolo verrà introdotto il protocollo di comunicazione principale del Web, HTTP, e verrà presentata la struttura dei messaggi HTTP di richiesta, inviati da un client (ad esempio un browser), e risposta, gestiti da un server (ad esempio il server web).

Introduzione

Nel precedente capitolo, in questo, e nei successivi affronteremo tutte le tecnologie alla base di Internet e del World Wide Web. Tuttavia, invece di affrontare questi temi in modo tradizionale, ovvero partendo dalla descrizione architetturale di Internet per poi presentare il Web, abbiamo voluto introdurre questi argomenti seguendo la traccia fornita da Tim-Berners Lee (l'inventore del Web) in un noto documento esplicativo a disposizione sul suo sito Web.

In particolare, l'obiettivo è quello di rispondere ad una semplice, seppur significativa, domanda:

- Cosa succede quando si clicca su un collegamento ipertestuale (o link) di una pagina Web?

In Figura 1 sono riassunte le tecnologie che abbiamo introdotto nel capitolo precedente e che ci hanno permesso di iniziare ad introdurre gli elementi principali per rispondere alla domanda appena presentata. In particolare, una volta che si clicca su un link di una pagina web visualizzata attraverso un browser, il browser recupera l'URL della risorsa celato dietro il link e richiede una copia di quella risorsa al server web che la ospita. Una volta che questa copia viene ricevuta dal browser, questi la visualizza (nel caso sia una pagina web, per esempio) o ne permette la memorizzazione all'interno del dispositivo da cui è partita la richiesta attraverso il browser usato.

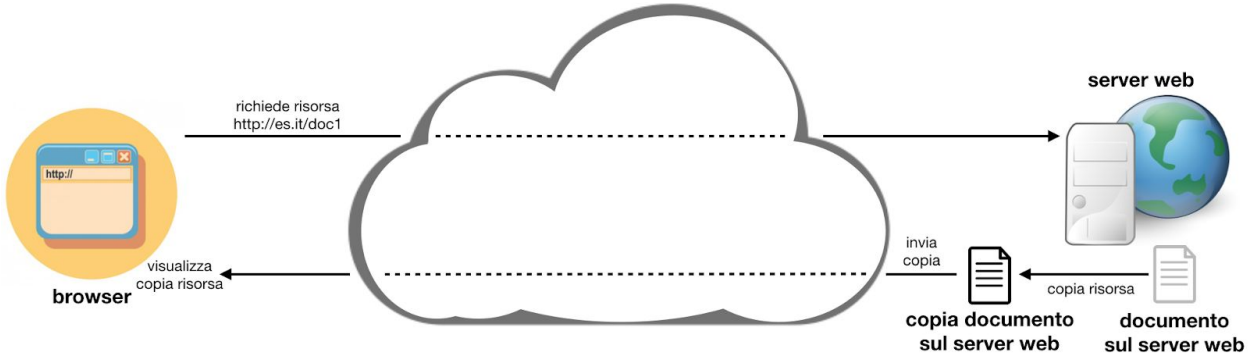

Figura 1. Riassunto delle tecnologie che abbiamo trattato nel capitolo precedente.

In questo capitolo approfondiremo meglio cosa siano i protocolli di comunicazione, in particolare quelli usati principalmente per comunicare richieste e informazione sul Web. Inoltre approfondiremo meglio il concetto di server web, delineandone le caratteristiche principali.

Che cos'è un protocollo di comunicazione

Formalmente, un protocollo di comunicazione è un **insieme di regole** che due entità, parte di un sistema di comunicazione, devono seguire **per scambiarsi informazioni**. Informalmente, può essere considerato come un linguaggio comune, nel senso proprio del termine, usato da due entità in modo da poter comunicare fra loro. Infatti, ogni protocollo definisce una specifica **sintassi** da seguire per costruire i messaggi, accompagnata da determinate regole interpretative del messaggio così da definirne anche la **semantica**. Inoltre, solitamente, un protocollo di comunicazione prevede anche dei meccanismi per **sincronizzare** la comunicazione e per **correggere e/o gestire eventuali errori** che possono intercorrere nello scambio dei messaggi.

Protocolli di varia natura, anche informali, sono utilizzati nella vita di tutti i giorni quando vogliamo, per esempio, comunicare con qualcuno. Per esempio, il diagramma in Figura 2 riassume le componenti principali del protocollo, strettamente informale, che due persone,

fluenti nella lingua italiana, utilizzano per scambiarsi delle informazioni verbalmente. In questo caso, mentre la sintassi della lingua è veicolata da opportune regole, e la semantica delle varie parole che compongono il messaggio è stata appresa nel tempo con l'apprendimento della lingua stessa, le altre componenti del protocollo derivano principalmente da regole sociali (ad esempio, il parlare uno alla volta) e dalle proprie conoscenze (ad esempio, riproponendo il senso di un messaggio usando differenti costrutti se è poco chiaro all'interlocutore) e dall'ambiente circostante (ad esempio, alzando la voce per farsi sentire meglio).

Figura 2. Esempio di protocollo di comunicazione tra due persone che usano l'italiano come lingua di comunicazione.

Nel corso di questo e dei successivi capitoli, analizzeremo alcuni dei protocolli di comunicazione più importanti per il funzionamento del Web e di Internet, a partire dal protocollo HTTP.

Come funziona l'Hypertext Transfer Protocol (HTTP)

L'Hypertext Transfer Protocol, o HTTP [Fielding et al., 1999], è un protocollo di comunicazione inizialmente sviluppato da Tim Berners-Lee per facilitare l'implementazione del Web, e poi migliorato ed esteso negli anni successivi – ad esempio con il protocollo HTTPS, che altri non è che un'estensione di HTTP che implementa un protocollo di comunicazione più sicuro sul Web, in particolare quando è necessario trasmettere dati di autenticazione come nomi utenti e password. HTTP è un protocollo basato su metodi di comunicazione di richiesta-risposta, tipico di molti protocolli (anche informali) di comunicazione – si pensi, ad esempio, al protocollo che regola le telefonate, dove un richiedente chiama (la richiesta) un ricevente che accetta la chiamata (la risposta) per iniziare la conversazione.

Nel caso di HTTP, un *client* (l'agente che fa la richiesta, ad esempio un browser) manda un **messaggio di richiesta** di una specifica risorsa, identificata da un URL, ad un *server* (l'agente che dovrebbe avere informazioni su quella specifica risorsa, ad esempio un server Web), che restituisce al client un **messaggio di risposta**. La composizione di questo messaggio di

risposta dipende da diversi fattori – ad esempio se la risorsa è o non è presente sul server, se è stata spostata in un'altra locazione, etc. – e, anche, dal tipo di richiesta effettuata dal client.

Le tipologie di richieste che possono essere effettuate da un *client* vengono definite grazie all'uso di uno specifico metodo, ovvero l'operazione di richiesta relativa all'URL coinvolto nella comunicazione. In pratica è come se, quando si comunica con qualcuno, si avesse la possibilità di scegliere tra alcune possibili operazioni che il ricevente deve eseguire, che possono andare dalla semplice richiesta di informazioni (ad esempio, quando chiamiamo la banca per sapere lo stato del nostro conto), ad azioni più operative che determinano l'esecuzione di una specifica azione da parte del ricevente (ad esempio, quando chiamiamo la banca per far emettere un bonifico).

Il metodo più usato nel Web è sicuramente *GET*, che permette di richiedere informazioni sulla risorsa definita dall'URL specificato. GET viene usato ogni qual volta richiediamo di visitare una specifica pagina web, per esempio come conseguenza di un click su un collegamento ipertestuale. Altri metodi largamente usati sono:

- *PUT* permette di creare sul server web contattato la risorsa specificata dall'URL utilizzato nella richiesta, e di associare le informazioni, relative ad essa, incluse nella richiesta effettuata;
- *DELETE* permette di rimuovere dal server web contattato tutte le informazioni relative alla risorsa specificata nella richiesta tramite l'URL;
- *POST* permette di specificare informazioni aggiuntive, incluse nella richiesta, ad una risorsa esistente che già risiede sul server.

Tuttavia, mentre GET è preponderante nel Web, visto che viene usato per richiedere una qualsiasi risorsa, gli altri metodi non sempre sono liberamente utilizzabili da tutti i client. Sta sempre al server web decidere quali di questi metodi possano essere usati nella comunicazione.

Tutti i metodi HTTP sono classificabili a seconda che siano conformi o meno a due caratteristiche specifiche, ovvero se sono *safe* e/o se sono *idempotenti*. Un metodo si dice *safe* quando viene usato solo per recuperare delle informazioni dal server web, senza cambiarne lo stato, ad esempio aggiungendo nuovi dati. Il metodo GET è un metodo *safe*, considerando che viene usato soltanto per richiedere informazioni relativamente ad una specifica risorsa, mentre PUT non lo è perché viene usato per creare una nuova risorsa sul server web. Invece, un metodo si dice *idempotente* se molteplici richieste effettuate allo stesso URL hanno lo stesso effetto, sul server web, che effettuare una sola richiesta. Anche in questo caso, il metodo GET, così come PUT e DELETE, sono metodi idempotenti, mentre POST non lo è, perché diverse richieste POST identiche possono di fatto generare comportamenti differenti.

A prescindere dal metodo usato per la comunicazione, tutti i messaggi di ogni richiesta e risposta veicolano due tipi di dati ben distinti: i metadati relativi alla comunicazione – come il

nome di *client* usato per effettuare la richiesta, il nome del server web che dovrebbe avere a disposizione la risorsa, etc. – ed eventuali dati (o contenuto) da accompagnare al messaggio. Per esempio, in una conversazione telefonica, il chiamante come prima cosa introduce chi è al ricevente (i metadati della comunicazione), per poi passare ad esplicitare il motivo della chiamata e le informazioni che si vorrebbero ottenere (il contenuto della comunicazione).

Considerando i messaggi consegnati utilizzando il protocollo HTTP, queste due tipologie di informazioni, ovvero i metadati e i dati, sono contenute in due specifiche sezioni del messaggio, rispettivamente chiamate *intestazione* (*header*) e *contenuto* (*payload*). Tipicamente, una qualunque richiesta o risposta ha sempre l'*header* specificato, mentre il *payload* può contenere informazioni come no. Per esempio, quando si richiede una pagina web attraverso un browser, viene effettuata una richiesta ad un certo URL contenente soltanto i metadati nell'*header*, mentre il *payload* è completamente vuoto. Al contrario, la risposta del server web contiene sia informazioni nell'*header* sia un vero e proprio documento nel *payload*, ovvero la pagina web che deve essere visualizzata sul browser richiedente, nel caso tale informazione sia a disposizione del server web a cui è arrivata la richiesta.

Tutte le risposte, oltre che a contenere metadati nell'*header* e, talvolta, dati nel *payload*, specificano anche un codice di stato che indica se la richiesta è andata a buon fine o se sono intercorsi dei problemi. Un codice di stato è un numero di tre cifre, la cui prima cifra (da 1 a 5) definisce la classe di risposta, mentre le altre due sono usate per identificare una specifica motivazione relativa alla risposta. Le cinque classi di risposta sono:

1. risposta informativa – la richiesta è stata ricevuta e correttamente compresa, e sono necessari alcuni passaggi aggiuntivi (attendere, cambiare protocollo, etc.) per poter ottenere le informazioni sulla risorsa richiesta;
2. successo – la richiesta è stata ricevuta, correttamente compresa, e accettata;
3. redirezione – il *client* deve effettuare specifiche azioni aggiuntive per completare la richiesta (ad esempio, cercare le informazione sulla risorsa ad un altro URL);
4. errore del *client* – la richiesta presenta un qualche errore lato *client* (ad esempio, non si ha il permesso di accedere alla risorsa, non è stata trovata alcuna informazione associata alla risorsa, etc.) e non può essere soddisfatta;
5. errore del *server* – il server web ha incontrato un qualche problema nel soddisfare la richiesta (ad esempio, server non è disponibile al momento a causa di un sovraccarico di gestione, non supporta il protocollo HTTP utilizzato dalla richiesta, etc.).

Mentre i codici della classe 1 sono solitamente abbastanza rari, i più comuni, in una comunicazione HTTP, sono i seguenti (tra parentesi le etichette esplicative ufficiali):

- codice 200 (OK) – la richiesta è stata soddisfatta con successo;
- codice 303 (See Other) – la risposta alla richiesta può essere trovata ad un altro URL;
- codice 400 (Bad Request) – la richiesta non può essere processata dal *server*;

- codice 403 (Forbidden) – la richiesta è valida, ma il *client* non ha i permessi necessari per accedere alle informazioni relative alla risorsa richiesta;
- codice 404 (Not Found) – le informazioni relative alla risorsa richiesta non sono state trovate;
- codice 500 (Internal Server Error) – un errore generico lato *server* non ha permesso il corretto processamento della richiesta;
- codice 503 (Service Unavailable) – il *server* è temporaneamente non disponibile.

Per riassumere: ogni qual volta un *client* manda una richiesta HTTP, utilizzando uno dei metodi HTTP a disposizione, relativamente ad una certa risorsa identificata da un URL, in realtà manda uno specifico messaggio contenente informazioni generali sulla richiesta (i metadati, contenuti nel campo *header* del messaggio) e, eventualmente, dei dati (il *payload* del messaggio) ad un *server* destinatario in qualche modo codificato all'interno dell'URL della risorsa richiesta. Il destinatario, letto il messaggio, restituisce risposta affermativa, con codice di stato 200, accompagnata dal relativo messaggio di risposta se può evadere la richiesta correttamente, altrimenti restituisce un opportuno messaggio di errore, così da rendere esplicito al mittente cosa non ha funzionato nella comunicazione.

Che cos'è un server web

Come anticipato nel precedente capitolo, un server web è un computer "speciale" che esegue un software specifico che permette di ricevere, gestire, e soddisfare richieste HTTP provenienti da un *client*, ad esempio un browser. Il nome del server web è incluso nell'URL che identifica la risorsa di cui si vuole ottenere informazioni, nella parte `host`, come introdotto nel capitolo precedente (ad esempio `it.wikipedia.org`). Tuttavia, la forma corretta e completa della richiesta HTTP mandata dal *client* non è quella semplicistica introdotta precedentemente col solo URL, ma è strutturata in modo preciso, in modo da identificare i vari componenti che ne fanno parte, incluso il nome del server web da contattare.

In particolare, quando un *client* (ad esempio, un browser) richiede informazioni su una specifica risorsa (ad esempio una pagina web) utilizzando l'URL relativo, la richiesta che viene davvero effettuata ha la seguente forma, dove ogni parte della richiesta è di fatto derivata dall'URL:

```
<metodo HTTP> <percorso> <protocollo>
HOST: <nome del server>
```

Ad esempio, quando si vuole accedere alla pagina web all'URL `http://it.wikipedia.org/wiki/Uniform_Resource_Locator`, la richiesta effettuata dal browser viene strutturata come segue:

```
GET /wiki/Uniform_Resource_Locator HTTP/1.1
HOST: it.wikipedia.org
```

Una volta che il server web riceve una richiesta fatta in questo modo, cerca localmente informazioni del documento indicate nel `percorso` della richiesta e, una volta trovate, le impacchetta in un nuovo messaggio di risposta, restituendole al richiedente.

Tuttavia, c'è un ulteriore aspetto molto importante che non è stato ancora discusso in questo capitolo. Seppur noi, utilizzatori di *client* come i browser, cerchiamo di accedere ad una risorsa a disposizione in un certo server web usando il nome di quest'ultimo, `it.wikipedia.org` nell'esempio precedente, in realtà il server web non è direttamente raggiungibile usando il suo nome, ma avviene una cosa molto simile a quello che succede con una comunicazione telefonica. In pratica, quando dobbiamo chiamare qualcuno col nostro cellulare, cerchiamo questa persona nella nostra rubrica attraverso il suo nome, e chiediamo al nostro cellulare di iniziare una chiamata telefonica. Tuttavia, quello che davvero fa il cellulare, è di prendere il numero di telefono associato a quella persona e di attivare la comunicazione mandando una richiesta di telefonata a questo numero. Ovviamente, in alternativa, si potrebbe anche digitare direttamente il numero di telefono della persona che si vuol chiamare se si conosce a memoria, senza necessariamente cercarla in rubrica. In più, se il numero chiamato è un numero fisso di casa, e in casa vivessero diverse persone, una volta che la telefonata è stata risposta, il mittente dovrebbe chiedere esplicitamente al ricevente di poter parlare con la persona di interesse.

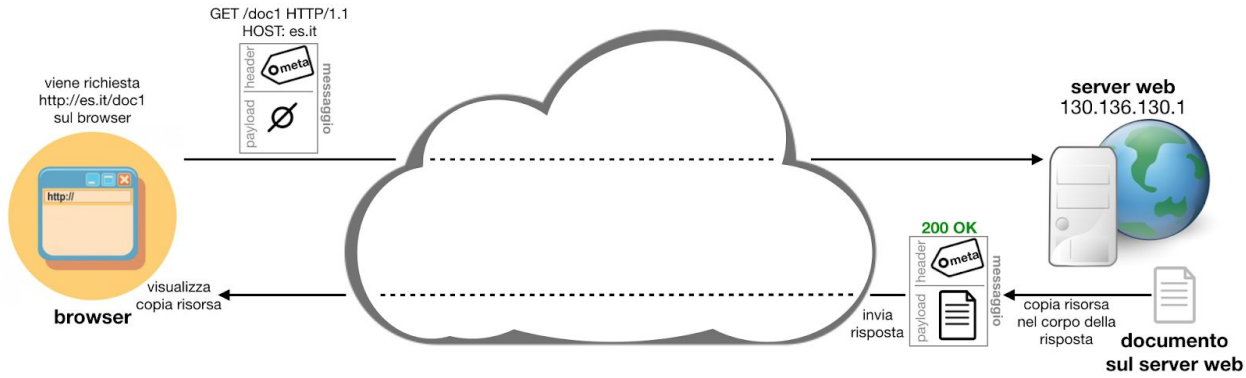

Figura 3. Riassunto delle tecnologie che abbiamo trattato in questo capitolo e nel precedente.

Questo scenario, molto naturale, si presenta in modo totalmente analogo anche quando un *client* richiede una pagina web ad un server web. Infatti, un qualunque server web è identificato in modo univoco da una sequenza di quattro numeri separati da punti, ad esempio `130.136.130.1`, ove ogni numero può essere un valore che va da 0 a 255. Questo numero si chiama indirizzo IP, dove "IP" è l'acronimo di "Internet Protocol", che discuteremo in dettaglio nel capitolo successivo. In pratica, ogni volta che si fa una richiesta HTTP, in qualche modo viene recuperato l'indirizzo IP del server web a cui mandare la richiesta partendo dal nome di questo indicato nell'URL – il campo `host` – per poi inviare il messaggio esplicitamente a quell'indirizzo, menzionando esplicitamente, come abbiamo visto in precedenza, qual è il nome del server a cui il messaggio deve essere rivolto – nell'esempio sopra `HOST:`

`it.wikipedia.org`. Figura 3 estende il diagramma in Figura 1 includendo tutte le tecnologie presentate in questo capitolo.

Bibliografia

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext Transfer Protocol -- HTTP/1.1 (Request For Comment No. RFC2616). Internet Engineering Task Force. https://doi.org/10.17487/rfc2616