

LA COMUNICAZIONE DEL WEB CON IL PROTOCOLLO HTTP



01

HTTP e il modello client-server

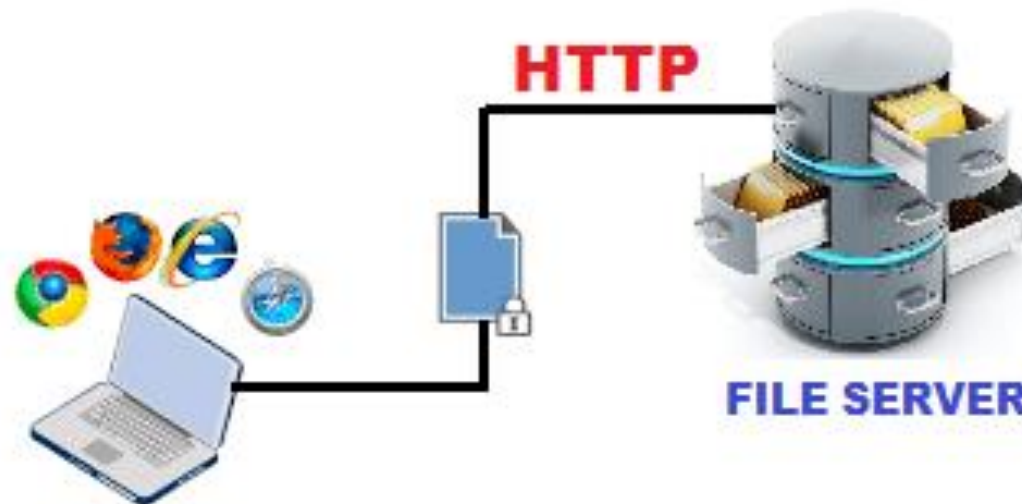
HTTP e il modello client-server

Il Web è basato sul modello client-server che ha due differenti elementi:

- Client;
- Server.

I **client** sono **elementi ATTIVI** (Web browser) che:

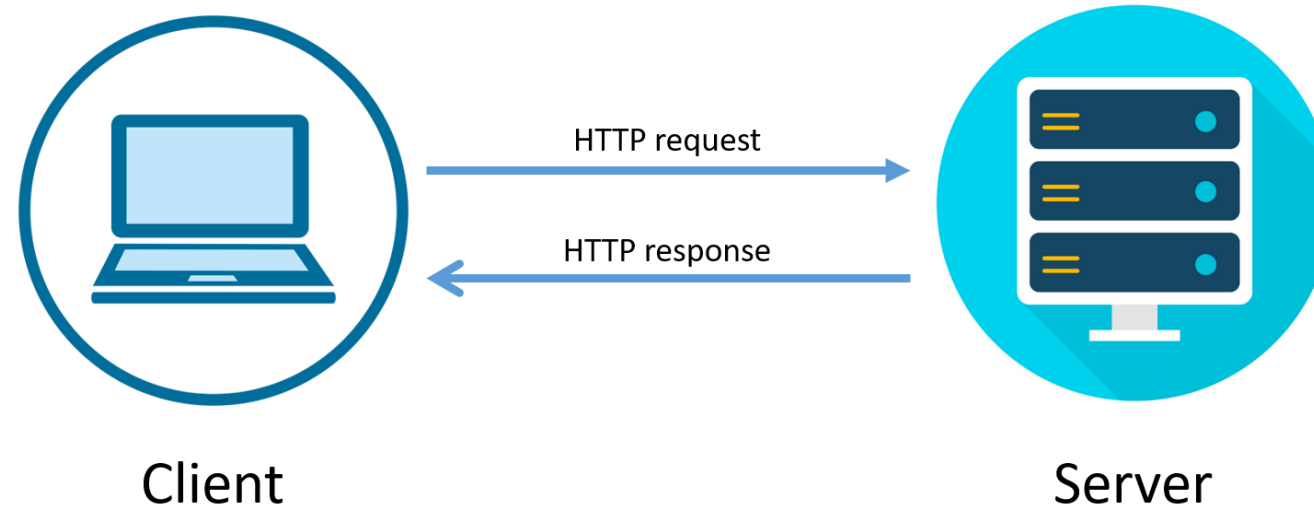
- utilizzano il protocollo HTTP per connettersi al server, secondo un modello a client attivo;
- usano un URL per identificare le risorse;
- richiedono pagine Web al server e ne visualizzano il contenuto.



HTTP e il modello client-server

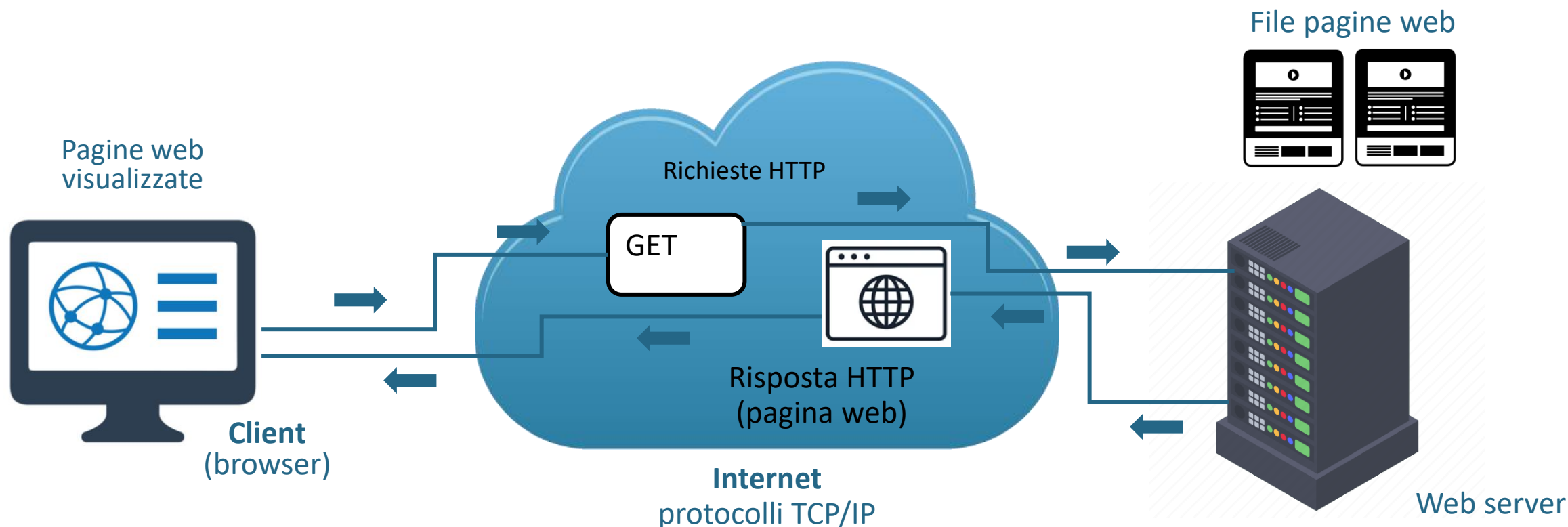
I server sono elementi PASSIVI (Web server o HTTP server) che:

- rimangono in ascolto di eventuali connessioni di nuovi client su una determinata porta TCP, secondo un modello a server passivo;
- utilizzando il protocollo HTTP per interagire con il client;
- forniscono ai client le pagine Web o le risorse richieste



HTTP e il modello client-server

I siti web sono ospitati su computer server che ricevono dai browser, in esecuzione su computer o dispositivi client, le richieste per singole pagine web che costituiscono il sito e alle quali rispondono inviando il contenuto delle pagine richieste. Il protocollo applicativo che regola il trasferimento delle pagine web dal web server al browser del client.



HTTP e il modello client-server

HTTP è un protocollo usato per la trasmettere risorse, non solo file. Un risorsa è indentificata da un URI o URL. Il tipo più comune di risorsa è un file, ma può anche essere il risultato di una query generata dinamicamente, l'uscita di uno script CGI, un documento disponibile in diversi linguaggi, o altro ancora.

Vediamo le definizioni del W3C (World Wide Web Consortium):

- **URI** (Uniform Resource Identifier): “set generico di nomi o indirizzi che rappresentano stringhe assegnate alle risorse”, viene solitamente utilizzato in XML, file di librerie di tag e altri file, come JSTL e XSTL, **IDENTIFICA UNA RISORSA**;
- **URL** (Uniform Resource Locator): “termine informale, utilizzato solo nelle specifiche tecniche, associato a popolari protocolli quali HTTP, FTP, mailto, ... ecc.”, serve principalmente per la ricerca di pagine Web, **CONSENTE DI INDIVIDUARE UNA RISORSA**.

https://ospite:pippo@www.istruzione.it:443/uploads/esercitiamoci.html/pagina=3#area_tutor

The diagram illustrates the components of the URL `https://ospite:pippo@www.istruzione.it:443/uploads/esercitiamoci.html/pagina=3#area_tutor`. Blue arrows point from each part of the URL to a corresponding label below it:

- `https` points to **Protocollo**
- `ospite:pippo` points to **Username** and **Password**
- `www.istruzione.it` points to **Host**
- `:443` points to **Porta**
- `/uploads/esercitiamoci.html/pagina=3` points to **Path** and **File**
- `#area_tutor` points to **Query** and **Fragment**

HTTP e il modello client-server

Protocollo	Identifica il protocollo dello strato dell'applicazione di cui si ha bisogno per accedere alla risorsa, in questo caso HTTP.
Username	Se il protocollo supporta il concetto di user name, questo componente ne fornisce uno che ha accesso alla risorsa.
Password	La password associata all'user name.
Host	Il sistema di comunicazione che ha la risorsa; per HTTP questo è il server Web.
Porta	La porta TCP che i protocolli dello strato dell'applicazione dovrebbero usare per accedere alla risorsa; a molti protocolli è stata assegnata una porta TCP specifica, per l'HTTP è 80.
Path	Il percorso attraverso un'organizzazione gerarchica tramite la quale la risorsa è localizzata, spesso una struttura di directory del file system o qualcosa di equivalente.
File	La risorsa stessa.
Query	Informazione aggiuntiva sulla risorsa o sul client.
Fragment	Una particolare ubicazione all'interno di una risorsa.

Il protocollo HTTP

HTTP è un protocollo di testo che fornisce il livello di trasporto a tutti i protocolli applicativi basati su di esso. Possiamo dire che il Web (**WWW**) è nato dall'insieme di diverse tecnologie, le principali sono **HTML**, **URL** e **HTTP**.

HTTP (acronimo di Hyper Text Transfer Protocol) è un protocollo che consente di inviare e ricevere le risorse Web, rappresentate da documenti, pagine o elementi di una singola pagina Web, da un host chiamato server a un altro chiamato client.

Il protocollo HTTP per comunicare utilizza sessioni, dove ciascuna sessione inizia stabilendo per prima cosa una connessione TCP al server, utilizzando di default la porta **TCP 80**, effettuando poi una richiesta (request) contenente un URL. In risposta, il server produce e restituisce il file richiesto e, nella situazione più semplice, chiude la connessione TCP immediatamente dopo.

Il protocollo HTTP

Fin dall'inizio il protocollo applicativo **HTTP** fu concepito per gestire risorse mediante comandi di richiesta, aggiornamento, creazione ed eliminazione. Per un browser che effettua richieste a un web server le risorse sono pagine che visualizza, ma oggi molte applicazioni e APP utilizzano il protocollo HTTP per richiedere, aggiornare, creare ed eliminare risorse software di qualsiasi tipo, come i record delle tabelle ed un database.

Il protocollo applicativo HTTP utilizza normalmente la porta TCP 80 con lo schema **http:**, ma per rendere riservata e non manipolabile la comunicazione client/server è possibile adottare il protocollo TLS (Transport Layer Security) per autenticare e cifrare la comunicazione effettuata con il protocollo di trasporto TCP: il protocollo di trasporto e quello applicativo e sulla sua base è stata realizzata la versione sicura del protocollo HTTP denominata HTTPS (HTTP Secure) che utilizza normalmente la TCP 443 con lo schema URL **https:**.

Il protocollo HTTP vs HTTPS

HTTP trasmette dati non crittografati, il che significa che le informazioni inviate da un browser possono essere intercettate e lette da terzi. Non era un processo ideale, quindi è stato esteso ad HTTPS per aggiungere un altro livello di sicurezza alla comunicazione. HTTPS combina richieste e risposte HTTP con tecnologia SSL e/o TLS.

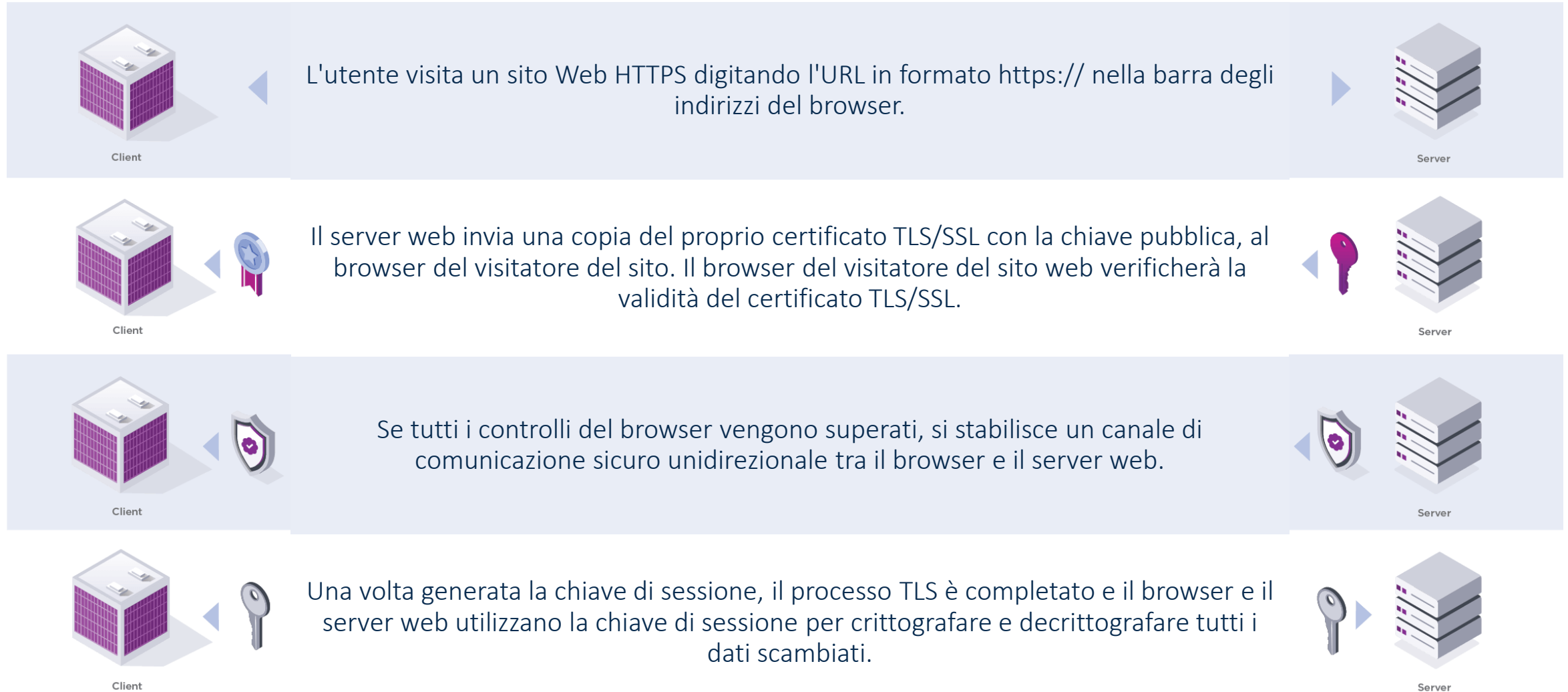
I siti Web HTTPS devono ottenere un certificato SSL/TLS da un'autorità di certificazione indipendente (CA). Il certificato SSL (Secure Sockets Layer) contiene informazioni crittografiche, quindi il server e i browser Web possono scambiarsi dati criptati.

Il protocollo HTTP vs HTTPS

Il processo funziona in questo modo:

1. L'utente visita un sito Web HTTPS digitando l'URL in formato `https://` nella barra degli indirizzi del browser.
2. Il browser prova a verificare l'autenticità del sito richiedendo il certificato SSL del server.
3. Il server Web invia il certificato SSL/TLS che contiene una chiave pubblica come risposta.
4. Il certificato SSL del sito Web dimostra l'identità del server. Una volta che il browser è soddisfatto del certificato, utilizza la chiave pubblica per crittografare e inviare un messaggio che contiene una chiave di sessione segreta.
5. Il server Web utilizza la propria chiave privata per decrittografare il messaggio e recuperare la chiave di sessione. Quindi utilizza la chiave di sessione per crittografare e inviare un messaggio di conferma al browser.
6. Ora, sia il browser che il server Web passano all'utilizzo della stessa chiave di sessione per scambiare messaggi in modo sicuro.

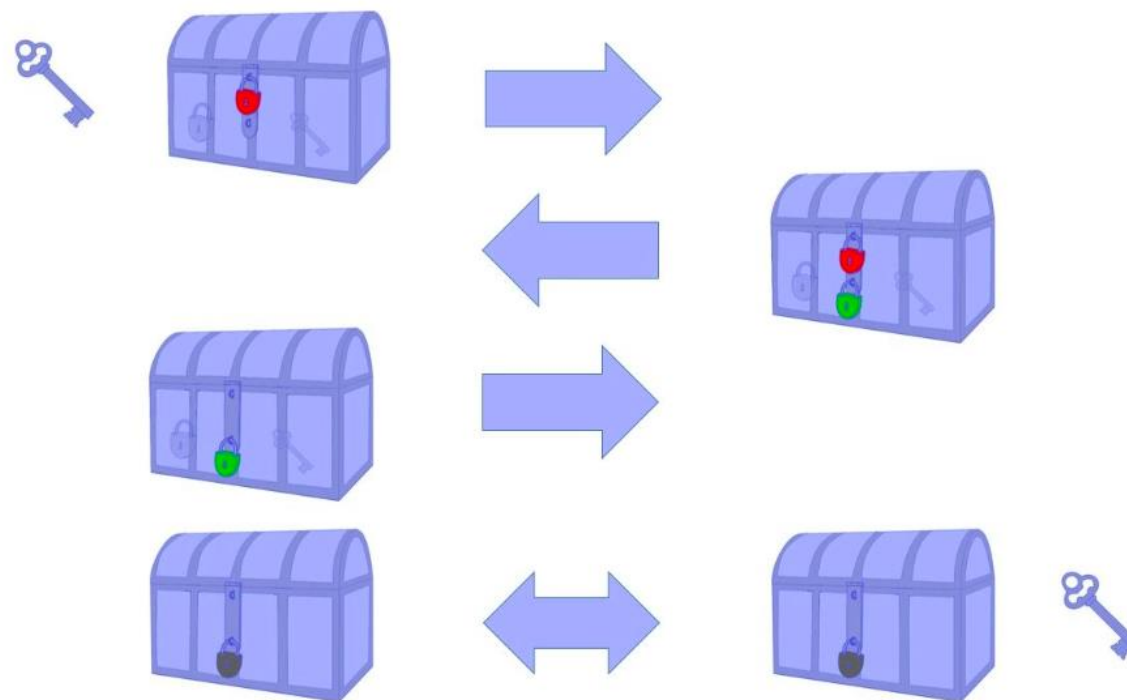
Il protocollo HTTP vs HTTPS



Il protocollo HTTP vs HTTPS

La **chiave di sessione** è una chiave utilizzata per implementare un canale sicuro tra mittente e destinatario mediante **cifratura simmetrica**. La particolarità di questa chiave è che essa è condivisa con tecniche che ne garantiscono la riservatezza. La chiave di sessione è mantenuta valida per un'intera sessione di lavoro prima che venga distrutta.

Le tecniche di scambio della chiave di sessione per il protocollo HTTPS afferiscono alla cifratura asimmetrica.

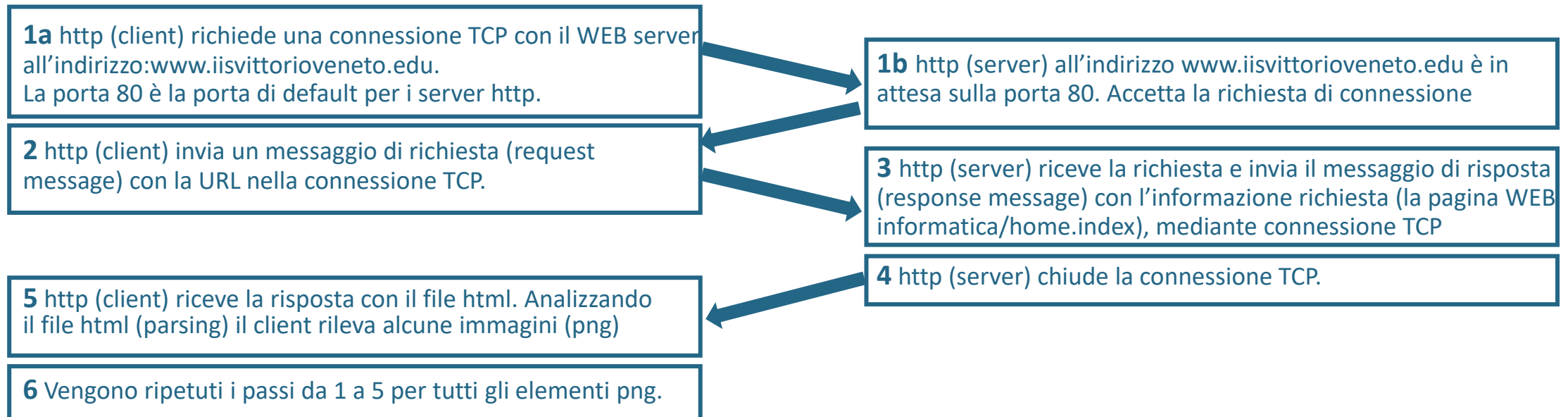


Conversazione client-server

Ogni conversazione tra client e server sul Web inizia con una richiesta (request) rappresentata da un messaggio di testo creato dal client in formato HTTP. Il client invia la richiesta al server, quindi attende la risposta (response).

ESEMPIO

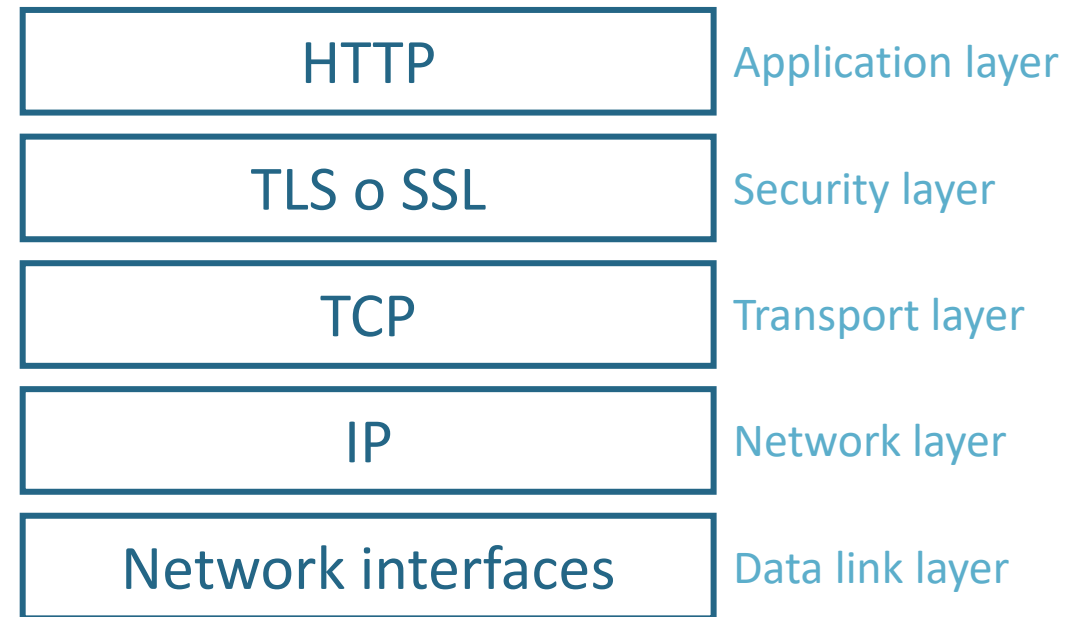
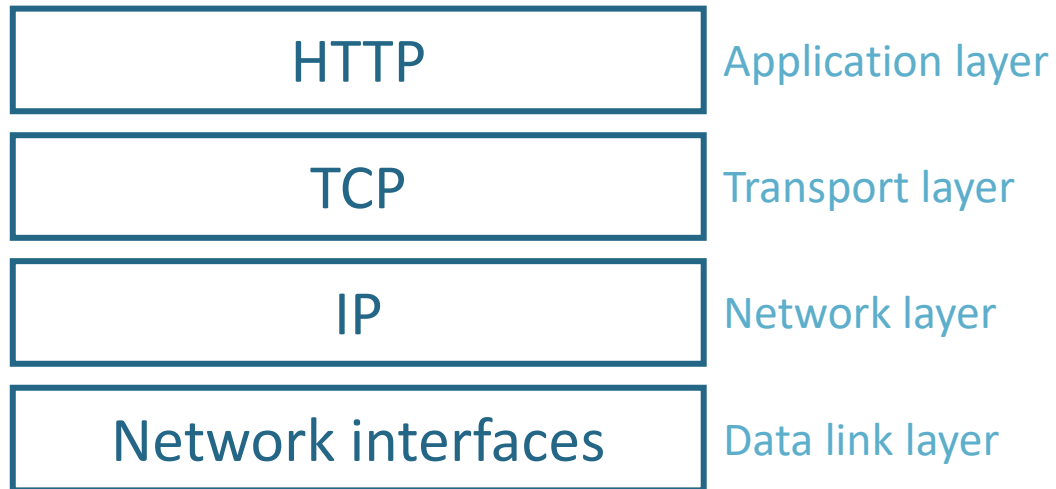
Ipotezziamo di volere richiedere una pagina composta da un file HTML e da alcune immagini in formato png all'indirizzo www.iisvittorioveneto.edu/informatica/home.index.



Tipi di connessioni

Il protocollo HTTP utilizza TCP come protocollo di trasporto per effettuare le connessioni tra client e server. La connessione HTTP tra client e server viene attivata dal client che attraverso lo scambio di tre segmenti TCP con il server.

Dopo tale scambio, il client può mandare la sua richiesta HTTP al server. Una volta soddisfatta la richiesta con un opportuno messaggio di risposta, è il server a chiudere la connessione tramite un ulteriore scambio di segmenti TCP.



Tipi di connessioni

Nella connessione HTTP 1.1, di **tipo permanente**, il server lascia aperta la connessione TCP dopo aver spedito la risposta. Le successive richieste e risposte fra client e server utilizzano sempre la stessa connessione. Una pagina Web, anche se formata da file distinti (immagini, file html ecc.) può essere inviata in una singola connessione TCP permanente.

Esistono due tipi di connessione permanente:

- connessione permanente incanalata;
- connessione permanente non incanalata.

Nella versione **non incanalata** il client passa una nuova richiesta solo quando la risposta alla precedente è stata ricevuta. Nella versione **incanalata** le richieste vengono via via aggiunte a una coda chiamata pipeline, mentre le risposte vengono processate e inviate nello stesso ordine delle richieste.

L'HTTP 1.1 utilizza di default la connessione incanalata: il client può fare richieste consecutive, effettuando una nuova richiesta non appena incontra un nuovo riferimento (metodo back-to-back).

I messaggi HTTP

Durante una connessione e una comunicazione HTTP **client** e **server** si scambiano **messaggi** di **richiesta** e di **risposta**, entrambi formati da:

- una **riga iniziale**;
- un'intestazione (header), non obbligatoriamente presente;
- una riga **vuota**;
- un corpo del messaggio (body), non obbligatoriamente presente

Messaggio di richiesta: HTTP Request

In generale la prima riga della request contiene la **versione** del protocollo **HTTP** utilizzato, mentre nelle righe successive vengono indicate gli **header** (intestazioni), rappresentate da diversi elementi, ciascuno dei quali composto da un nome, seguito dai due punti (:) e da un valore, dove ogni riga rappresenta un distinto header. Gli header più comuni che possiamo trovare nelle richieste sono:

- la versione del browser che prende il nome di User-Agent;
- l'host presente nell'URL;
- il parametro (per i tipi di documento che supportano il tipo MIME).

MIME (Internet media type) è un dato formato da due componenti che identifica la classe e il formato di ogni tipo di file. Per esempio text/plain audio/mpeg text/csv video/mp4 sono alcuni dei formati disponibili.

Gli header si concludono con una riga vuota, la quale può essere seguita da un qualunque dato da inviare al server. La dimensione massima del dato inviato è stabilito dall'header Content-Length. Generalmente HTML utilizza questo spazio per l'invio dei campi dei form.

Messaggio di richiesta: HTTP Request

Una HTTPrequest è un messaggio testuale inviato dal client al server HTTP ed è formato da tre elementi:

- riga di richiesta
- intestazione HTTP (header),
- corpo del messaggio (messagebody),

secondo la sintassi seguente:

<code><Method></code>	<code><URI></code>	<code><Version></code>	<code>//</code>	<code>riga di richiesta</code>
<code>[Header]</code>			<code>//</code>	<code>intestazione</code>
<code>CRLF</code>			<code>//</code>	<code>riga vuota</code>
<code>[Body]</code>			<code>//</code>	<code>corpo del messaggio</code>

CRLF indica l'invio a capo per ottenere una riga vuota e si ottiene con la sequenza di caratteri ASCII:

- CR = Carriage Return,
- LF = Line Feed.




Riga di richiesta

La riga di richiesta contiene:

- il metodo (Method) che può essere:
 - GET,
 - DELETE, GET, HEAD, LINK, POST, PUT, UNLINK,
 - CONNECT, DELETE, GET, HEAD, POST, PUT, OPTIONS, TRACE;
- l'URI è l'identificativo di risorsa richiesta al server;
- la versione (Version) può assumere i valori HTTP/1.0 o HTTP/1.1

ESEMPIO

Un esempio di riga di richiesta HTTP è la seguente:

GET	/percorso/file.html	HTTP/1.1
		
Metodo	URL	Versione

Intestazione HTTP (header)

L'**intestazione HTTP** (header) contiene tutte le informazioni necessarie per l'identificazione del messaggio. È formata da diverse righe, ciascuna delle quali rispetta il formato:

```
nome_header : valoreCRLF
```

e sono classificate in:

- **intestazioni generali** (si applicano solo al messaggio trasmesso e si applicano sia ad una richiesta che ad una risposta, ma non necessariamente alla risorsa trasmessa);
 - Ad es., Date: per data ed ora della trasmissione
- **intestazioni della richiesta** (impostati dal client per specificare informazioni sulla richiesta e su se stesso al server);
 - Ad es., User-Agent: stringa che descrive il client che origina la richiesta;
- **intestazioni del corpo dell'entità** (forniscono informazioni sul corpo del messaggio, o, se non vi è corpo, sulla risorsa specificata);
 - Ad es., Content-Type: il tipo MIME

Corpo del messaggio (Message body)

Il corpo del messaggio (message body) di richiesta contiene i dati trasportati.

ESEMPIO

Request line
(GET, POST, HEAD
commands)



GET / HTTP/1.1

Header lines

Accept: image/gif, image/jpeg, image/png, */*

Accept-Language: it

Accept-Encoding: gzip, deflate

Host: ww.magistricumacini.it

If-Modified-Since: Fri, 14 Mar 2018 10:54:03 GMT

User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; windows NT 5.0)

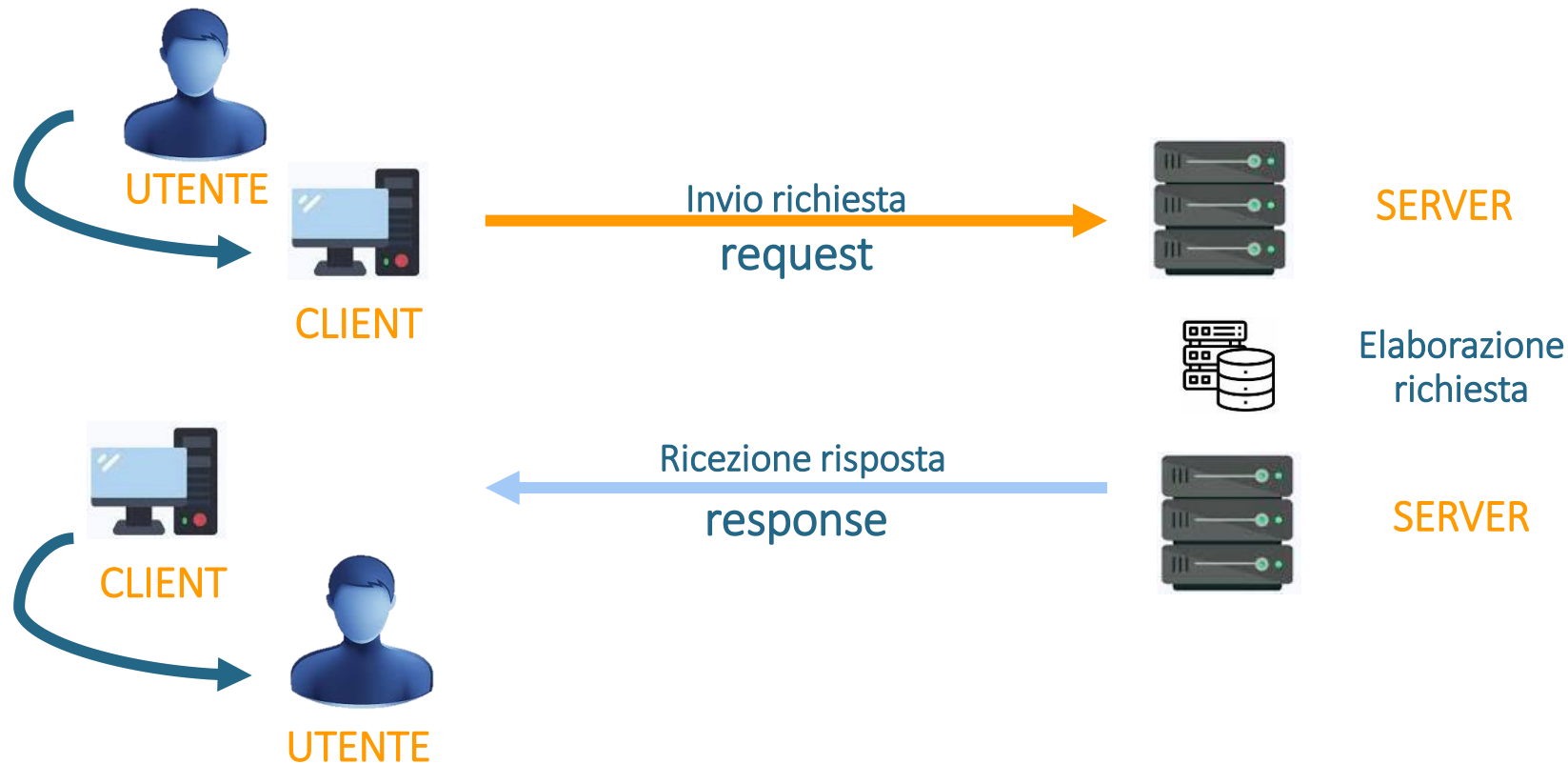


Indicatori la fine del
messaggio

(carriage return, line feed)

Messaggio di risposta: HTTP Response

Dopo che il server HTTP ha ricevuto una richiesta da un client, intraprende le azioni necessarie a fornire la risorsa richiesta. Queste possono includere l'esecuzione di uno script in tecnologia server-side. Fatto ciò, il server Web risponde al client con una risposta HTTP.



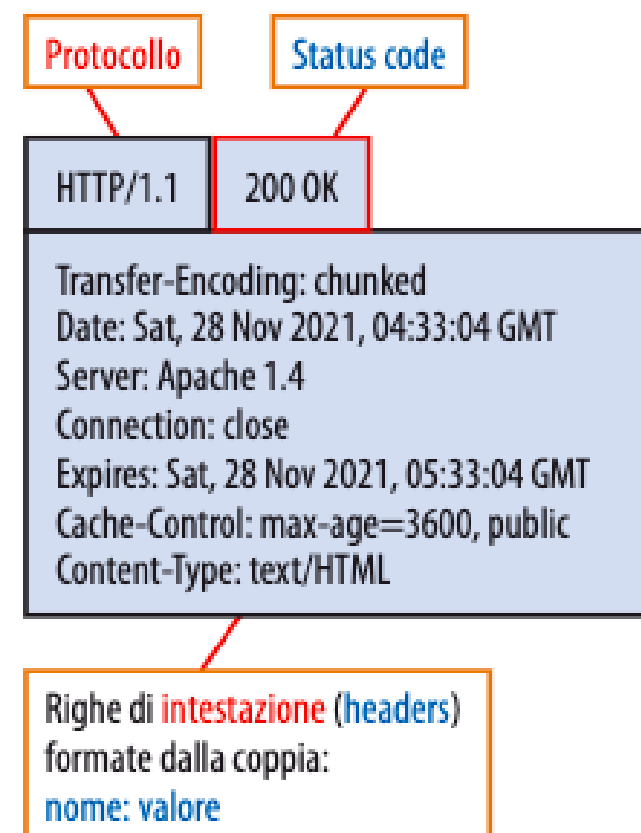
Messaggio di risposta: HTTP Response

La response (risposta) HTTP è organizzata in maniera analoga rispetto a una richiesta. L'unica differenza è che **le risposte iniziano con una riga di stato** al posto della riga di richiesta.

La response è formata da:

- una riga iniziale con versione protocollo HTTP e stato;
- un'intestazione (header) facoltativa;
- un corpo (body) facoltativo.

Nella seguente figura vediamo un'intestazione tipica di una response.



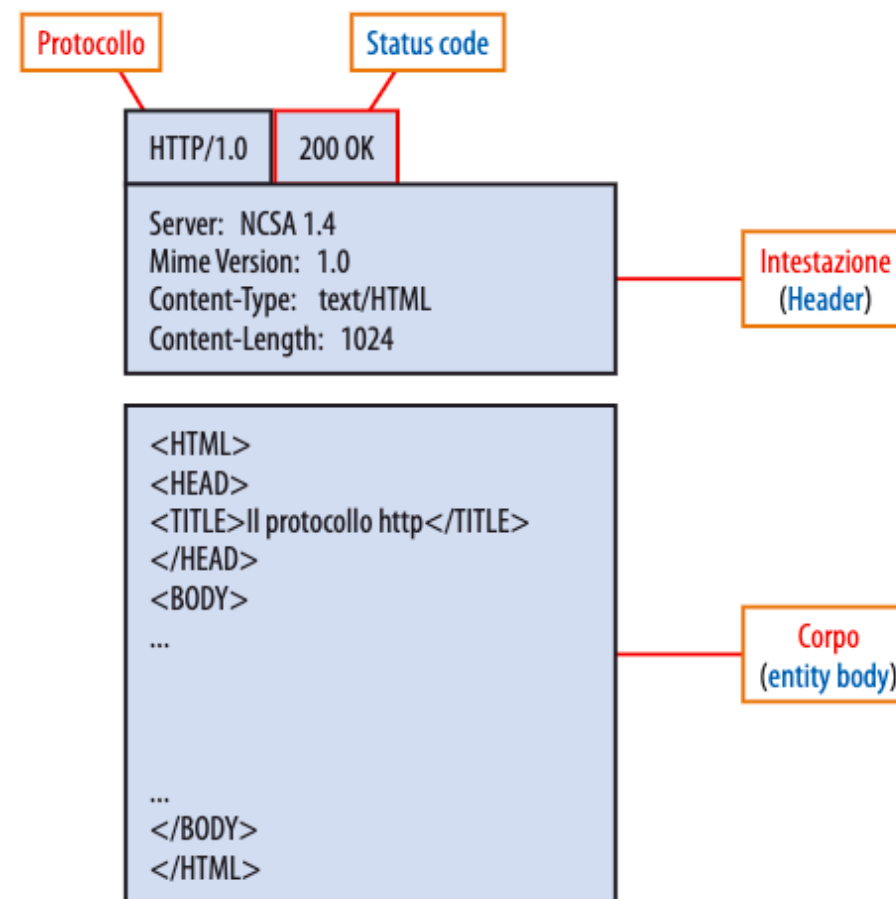
Header HTTP

Gli header della response sono formati, come nel caso della richiesta, da insiemi di coppie, formate da un nome separato dal valore dai due punti (:), che specificano le caratteristiche del messaggio.

- **Header generali della trasmissione**
 - MIME-Version: la versione MIME usata per la trasmissione, normalmente 1.0
 - Transfer-Encoding: indica la compressione dei dati trasmessi e si riferisce a tutto il messaggio.
- **Header relativi all'entità trasmessa**
 - Content-Type: indica il MIME e specifica se si tratta di un'immagine PNG, un filmato MPG ecc.;
 - Expires: indica la data di validità della risorsa.
- **Header riguardo la richiesta effettuata**
 - User-Agent: indica la versione e il tipo di browser oltre alla versione e al sistema operativo del client
 - Host: indica il nome di dominio del server e la porta TCP sulla quale il server è in ascolto.
- **Header della risposta generata**
 - Server: una stringa che descrive il server (tipo, sistema operativo e versione);
 - WWW-Authenticate: indica il tipo di autenticazione e le credenziali per autenticare un utente

Body della risposta

La sezione **body** del messaggio di risposta (entity body response) contiene in genere la pagina HTML richiesta dal client. L'entity body è quasi sempre presente nella risposta. La figura mostra un esempio di risposta di un server a una richiesta di un client.



Metodi (verbi) HTTP

Nella conversazione HTTP, la prima riga di intestazione di una request contiene un elemento chiamato metodo o verbo HTTP, che viene sempre digitato tutto in maiuscolo, per esempio:

```
GET / HTTP/1.1
```

In questo caso viene richiesto il metodo GET durante l'invio del messaggio di request. In questo secondo caso viene invece richiesto il metodo DELETE:

```
DELETE /prove/risorsa HTTP/1.1
```

Il metodo di richiesta è uno dei più importanti attributi di una richiesta HTTP, in quanto rappresenta l'“intenzione” presente nella richiesta del client. Sebbene molti metodi siano usati raramente nella pratica, sono tutti importanti per scopi diversi.

Metodi (verbi) HTTP

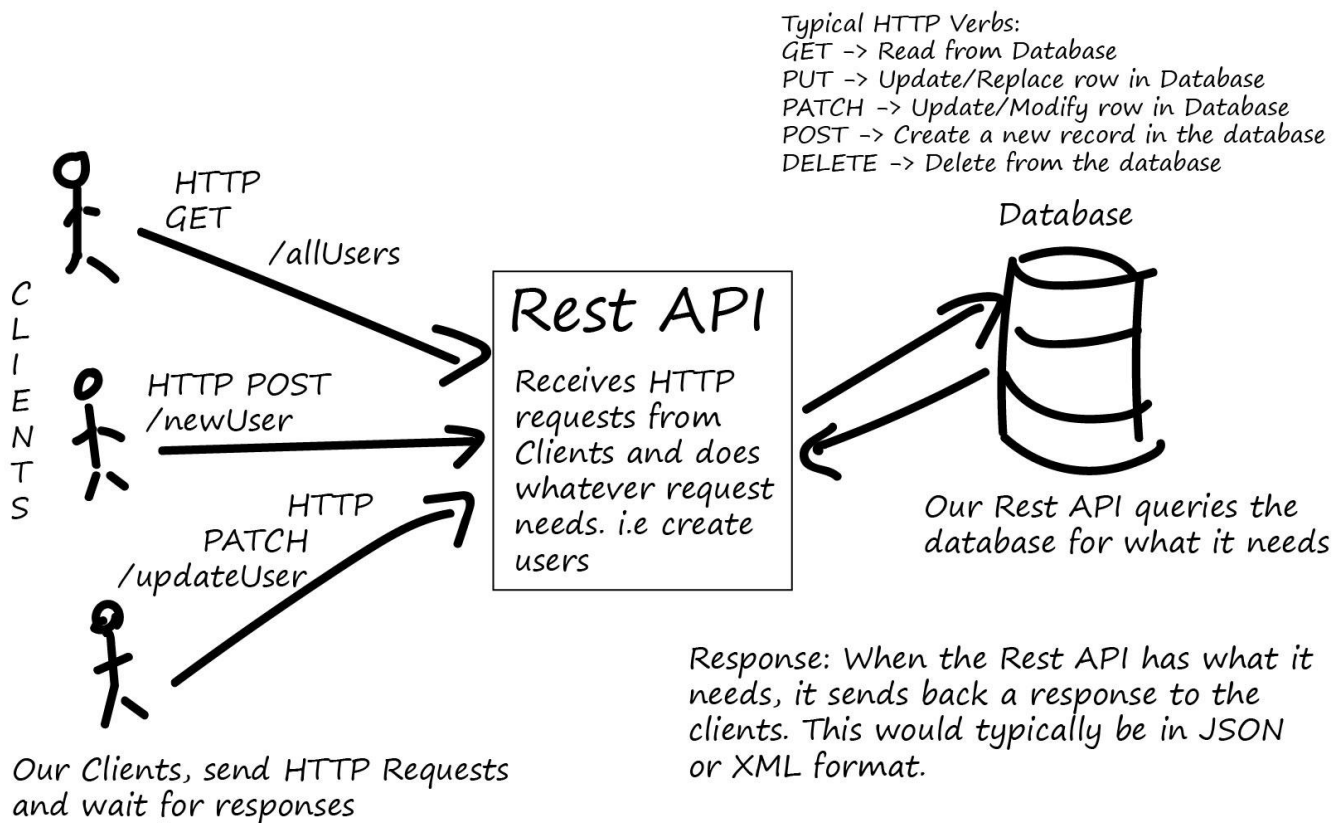
METODO	VERSIONE HTTP	MODIFICHE IN HTTP/1.1	DESCRIZIONE
GET	1.0 e 1.1	Richiesta di parti di entità	Richiede un file al server. La risposta è composta da vari header separati in genere rappresentato dal file HTML che contiene la pagina richiesta. Tra i dati della risposta si troverà il protocollo usato (HTTP/1.1) e lo status code che indica l'esito della richiesta.
HEAD	1.0 e 1.1	Uso di header di richiesta condizionali	Richiede solo l'header, senza alcun percorso di file. Utilizzato solo in ambito di fase di testing per pagine Web dinamiche.
POST	1.0 e 1.1	Gestione di connessione, trasmissione di messaggio	Invia informazioni all'URL specificato. Le informazioni vengono elaborate dal server.

Metodi (verbi) HTTP

METODO	VERSIONE HTTP	MODIFICHE IN HTTP/1.1	DESCRIZIONE
PUT	1.1 (1.0 non standard)		Esegue l'upload di un file sul server, creandolo o riscrivendolo, con il nome indicato e i contenuti specificati nella parte che segue gli header.
DELETE	1.1 (1.0 non standard)		Cancella una risorsa, generalmente rappresentata da un file, sul server. L'utente con cui è in esecuzione il Web server deve poter avere permessi in scrittura sul file indicato e il server deve essere configurato opportunamente (permesso di cancellazione).
OPTIONS	1.1	Estensibilità	Richiede l'elenco dei metodi concessi dal server, per esempio: <div>OPTIONS * HTTP/1.0</div>
TRACE	1.1	Estensibilità	Traccia una richiesta, visualizzando come viene trattata dal server, per esempio <div>TRACE * HTTP/1.0</div>

Metodi (verbi) HTTP

I più importanti metodi HTTP necessari per la conversazione attraverso applicazioni Web server-side (API) di tipo **RESTful** sono GET, POST, PUT e DELETE. Le API per HTTP consentono di eseguire operazioni sui dati presenti sui server.



Metodi (verbi) HTTP

Le applicazioni RESTful (REpresentational State Transfer) consentono di effettuare operazioni da remoto sui dati presenti nei server, che lo permettono, sfruttando i metodi del protocollo HTTP. Le operazioni del tipo CRUD (Create, Retrieve, Update, Delete), cioè crea, recupera, aggiorna e cancella, possono essere così descritte:

- chiedere dati al server (GET);
- creare dati sul server (POST);
- modificare o sostituire i dati sul server (PUT);
- cancellare un oggetto sul server (DELETE).

Non hanno memoria di stato nelle richieste.

La codifica URL

Nella codifica URL, chiamata anche URL encode, i dati (campi modulo) di un form HTML vengono codificati per essere impacchettati in una trasmissione GET o POST.

La URL encode è una procedura che prepara opportunamente una stringa di dati, codificandola, in modo che venga riconosciuta durante la trasmissione HTTP con metodo GET o POST. In altre parole, questo tipo di codifica viene utilizzata nelle QueryString per inserire, all'interno delle URL, delle stringhe di testo in modo sicuro e conforme allo standard previsto per le URL.

La codifica URL viene applicata anche nel caso in cui la richiesta venga generata, per esempio, da un link HTML (tag <a href>) e non solo da campi di un modulo (tag <form>).

La codifica URL

La stringa da inviare è sempre formata da una serie di coppie nome campo e valore: possiamo paragonare il nome campo a una specie di variabile e il valore al dato in essa contenuto; le operazioni da fare sono le seguenti:

1. convertire tutti i caratteri non sicuri presenti nella stringa in simboli formati dalla percentuale e dal codice ASCII relativo: %xx, dove xx è il valore ASCII del carattere, in esadecimale. I caratteri non sicuri includono =, &, %, @, spazio, +, e altri caratteri non stampabili;
2. eliminare gli spazi vuoti sostituendoli con il carattere %20;
3. unire i nomi e valori con uguale (=) e separare i nomi con ampersand (&), per esempio:

nome1=valore1&nome2=valore2&nome3=valore3 ...ecc.

La codifica URL

tabella che riepiloga i principali caratteri di URL encode:

CARATTERE	CODIFICA URL ENCODE
	%20
"	%22
*	%2A
0	%30
3	%33
6	%36
9	%39
@	%40

CARATTERE	CODIFICA URL ENCODE
#	%23
\$	%24
/	%2F
1	%31
4	%34
7	%37
<	%3C
[%5B

CARATTERE	CODIFICA URL ENCODE
%	%25
&	%26
2	%32
5	%35
8	%38
>	%3E
]	%5D

tabella che riepiloga i principali caratteri di URL encode:

La codifica URL

Un form contiene un campo chiamato utente che contiene il valore Luca e un campo chiamato societa che contiene Rossi & Martini, vediamo come viene codificata la stringa secondo URLEncode:

```
utente=Luca&societa=Rossi%26Martini
```

Le rappresentazioni HTTP

Come abbiamo visto client e server HTTP si scambiano informazioni riguardanti risorse identificate da un indirizzo (URL o URI): sia la richiesta sia la risposta contengono una rappresentazione della risorsa.

La rappresentazione della risorsa HTTP è un'informazione che riguarda non solo la risorsa intesa come indirizzo di localizzazione, bensì il formato e lo stato della risorsa attuale e nella sua evoluzione futura. I due elementi cardine della rappresentazione sono l'intestazione (header) e il corpo (body) del messaggio.

L'header HTTP contiene informazioni chiamate metadati, definite strettamente dalle specifiche HTTP: contengono testo e devono essere formattate, come abbiamo visto, in maniera specifica. Il body HTTP contiene informazioni che rappresentano dati in qualunque formato (testo semplice, immagini, HTML, XML, JSON ecc.). La risposta HTTP deve specificare il tipo di contenuto del body, attraverso l'intestazione, nel campo Content-Type, come per esempio in questo caso in cui viene richiesta una risorsa JSON:

```
Content-Type: application/json
```


I codici di stato

I codici di stato, o status code, sono dei codici restituiti dai server HTTP per indicare al client l'esito di una richiesta. Sono stati definiti dall'IETF.

L'RFC 2616 elenca circa 50 codici di stato che possono essere restituiti dai server HTTP, nella response. La maggior parte degli stati di questo elenco non corrisponde con esattezza al comportamento delle moderne applicazioni Web, alcuni codici di risposta sono stati infatti pensati per usi futuri.

I codici di stato

Lo status code è un numero formato da tre cifre, di cui la prima indica la classe della risposta e le altre due la risposta specifica. Esistono cinque classi principali di codici di stato, suddivisi in base alla tipologia di risposta, vediamoli riassunti nel seguente elenco dove sono stati suddivisi per categoria:

1. **codici da 100-199 (Information):** questi codici di risposta forniscono informazioni temporanee alla richiesta, durante il suo svolgimento e a partire dall'HTTP versione 1.0 viene sconsigliato il loro utilizzo;
2. **codici da 200-299 (Successful):** questo intervallo di codici di stato indica il completamento di una richiesta con successo;
3. **codici da 300-399 (Redirection):** questi codici vengono utilizzati per comunicare una varietà di stati che non indicano un errore, ma che richiedono un trattamento particolare da parte del browser;
4. **codici da 400-499 (Client error):** questo intervallo di valori viene utilizzato per indicare le condizioni di errore provocate dal comportamento del client;
5. **codici da 500-599 (Server error):** è una classe di messaggi di errore, prodotti come risposta a problemi verificatisi sul server.

I codici di stato

CODICI DI STATO SPECIFICI	DESCRIZIONE
100 Continue	Il client può chiedere al server di accettare una richiesta prima di inviare l'intero corpo del messaggio. L'intestazione Expect: 100-Continue chiede al server di segnalare la sua accettazione: quando il client riceve lo status code 100 dal server, continua la trasmissione inviando il resto della richiesta.
200 OK	È la risposta di default a una richiesta GET o POST che indica un esito positivo. Il browser visualizza i dati ricevuti immediata-mente dopo o li processa.
201 Created	Questo codice indica che la richiesta era valida e che una risorsa è stata creata. Viene utilizzato per confermare il successo di una richiesta PUT o POST.
204 No content	Questo codice identifica un esito positivo, quando non sono attesi dati di risposta, e blocca la navigazione sull'URL che l'ha generato, mantenendo l'utente sulla pagina di origine.
206 Partial Content	Il codice viene generato dal server in risposta a richieste di blocchi da parte del client. Il client deve già essere in possesso degli altri "blocchi" (chunk) del documento e esamina l'header di risposta Content-Range per ricomporlo prima di effettuarne l'elaborazione.
301 Moved Permanently 302 Found 303 See Other	Questi codici comunicano al client di ripetere la richiesta a un nuovo server, specificato nell'header Location presente nella response: il client sostituisce POST con GET, elimina il payload e inoltrando nuovamente la richiesta.

I codici di stato

CODICI DI STATO SPECIFICI	DESCRIZIONE
400 Bad request	Attraverso i codici 400 il server informa che non è in grado di processare la richiesta per qualche motivo non specificato: il payload della risposta in genere illustra il problema e generalmente viene gestito dal client come una risposta 200.
401 Unauthorized	Questa risposta indica che l'utente deve fornire delle credenziali di autenticazione a livello di protocollo HTTP per accedere alla risorsa. In genere a questo punto il browser chiede all'utente le informazioni per fare il login e visualizza il corpo della risposta solo se il processo di autenticazione non va a buon fine.
403 Forbidden	L'URL richiesto esiste ma non può essere scaricato per ragioni diverse dall'autenticazione HTTP, per esempio, permessi di accesso al filesystem non sufficienti, una regola di configurazione sul server che vieta l'accesso alla risorsa, credenziali insufficienti come, per esempio, quelle rappresentate da cookie non validi o da indirizzi IP non validi. Questa risposta viene generalmente mostrata all'utente.
404 non trovato	Il codice 404 è forse il più noto status code di errore, in quanto indica che l'indirizzo (URL) richiesto non esiste. Questa risposta viene generalmente mostrata all'utente.
405 metodo non permesso	Il metodo utilizzato non è supportato per la risorsa in questione.
500 errore interno al server	La risposta 500 è usata quando il processo fallisce dal lato server, in circostanze impreviste, che causano l'insorgere dell'errore.

cURL

Un altro utile metodo per prendere confidenza con l'HTTP è di usare un client dedicato, come per esempio cURL, scaricabile dall'indirizzo <https://curl.haxx.se/>.

cURL è un software gratuito e open source definito dai suoi stessi autori come “command line tool and library for transferring data with URLs”.

cURL viene utilizzato nelle righe di comando o negli script per trasferire i dati in tutti i tipi di dispositivi, dal PC al tablet, dai televisori alle stampanti, dai router alle apparecchiature Audio EDD.

cURL viene utilizzato per effettuare i test dei siti Web, sia semplici come quelli composti da poche pagine che noi realizzeremo, sia estremamente complessi come Facebook o YouTube.

cURL è un software da riga di comando per Shell/DOS che, oltre a HTTP, interagisce con svariati protocolli, come FTP e TELNET.

cURL

Con cURL è possibile effettuare molte operazioni:

- recuperare una pagina Web e il suo contenuto;
- inviare in modo fittizio i dati di un form;
- autenticarci in modo fittizio in un'area riservata;
- connetterci in modo generico a un Web service



02

Le applicazioni Web e il modello client-server

Applicazioni Web: generalità

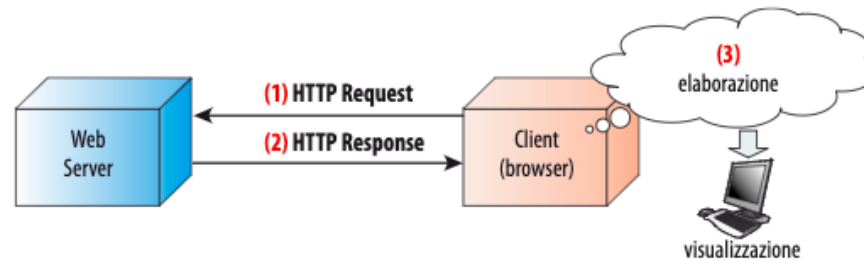
Con il termine di applicazione Web viene definito il software sviluppato e utilizzato attraverso tecnologie Web e linguaggi specifici. Sintetizziamo due concetti fondamentali che stanno alla base delle applicazioni Web:

1. tecnologie client-side e server-side;
2. linguaggi di mark-up e linguaggi di programmazione.

Tecnologie del Web

Possiamo distinguere le tecnologie del Web in due gruppi, in base al posto in cui avvengono le elaborazioni.

1. Tecnologie client-side: sono le strutture tecnologiche in cui l'elaborazione avviene sul client, tipicamente nel browser; possiamo schematizzare le tre fasi dell'elaborazione in:



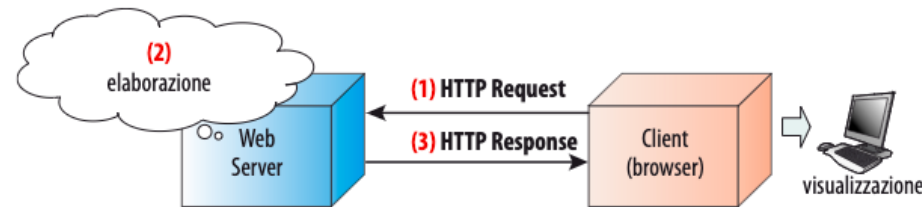
Per visualizzare una pagina che utilizza una tecnologia client-side (che già conosciamo essere HTML, Java-Script ...) potremmo anche non utilizzare un Web server, memorizzando e, quindi, aprendo la pagina WEB e fornendo al browser il path sul file system locale:



Inoltre, se chiediamo al browser di visualizzare il codice sorgente della pagina, le parti di codice che utilizzano tecnologie client-side sono visibili.

Tecnologie del Web

2. Tecnologie server-side: sono le strutture tecnologiche in cui l'elaborazione avviene sul server, tipicamente nel Web server; possiamo schematizzare le tre fasi dell'elaborazione in:



Per visualizzare una pagina che utilizza una tecnologia server-side (scritta in linguaggi dinamici come PHP, Java Servlet,..) abbiamo bisogno di un Web server che elabori il codice della pagina ed è, quindi, necessario connettersi e richiedere la pagina tramite un URL:



Se, invece, chiediamo al browser di visualizzare le parti di codice che utilizzano tecnologie server-side queste NON sono visibili perché il server ha elaborato il codice (che è memorizzato sullo stesso server) e quello che vediamo sul nostro browser è il risultato di tale elaborazione.

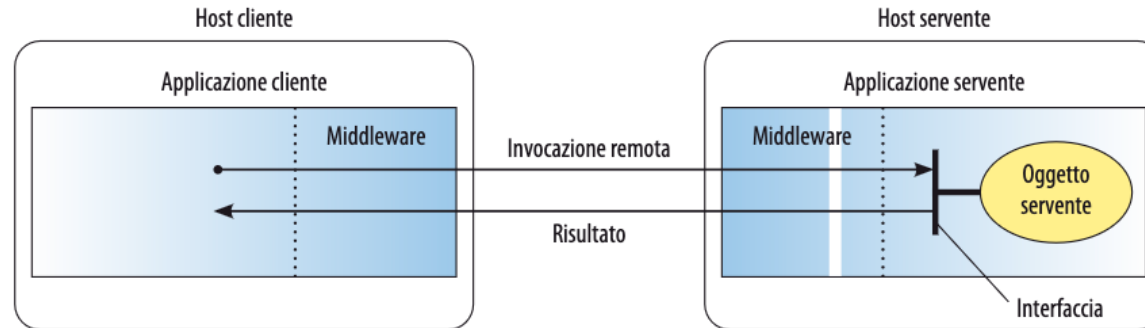
Linguaggio del Web

Nel Web vengono utilizzate due tipologie di linguaggi: i linguaggi di mark-up e i linguaggi di programmazione.

1. I linguaggi di mark-up servono a scrivere documenti strutturati (“formattati”), dove un documento “formattato” è un contenuto generalmente testuale corredato di indicazioni (tag) che ne definiscono la struttura (e spesso la visualizzazione), come il linguaggio HTML, XML.
2. I linguaggi di programmazione servono a scrivere programmi: un programma è una sequenza di istruzioni, come per esempio Java e PHP.

Il modello client-server

Il modello client-server è costituito da un insieme di host che gestiscono una (o più) risorse, i server, e da un insieme di client che richiedono l'accesso ad alcune risorse distribuite gestite dai server. Inoltre ogni processo server può a sua volta diventare client per richiedere accesso ad altre risorse gestite da altri (processi) server.



Per essere precisi, non sono gli host a essere server o client ma i processi che sono in esecuzione su di essi, dove come processo si intende un programma in esecuzione: dato che su un host possono essere in esecuzione più processi, un host può essere contemporaneamente sia client che server.

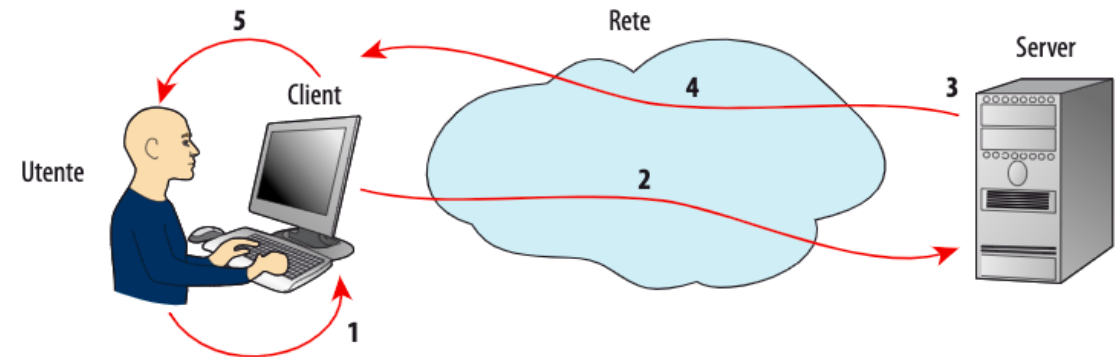
Il modello client-server

Descriviamo il modello client server anche analizzando la sua evoluzione che oggi gli permette di gestire applicazioni molto complesse (applicazioni di tipo enterprise) che hanno le seguenti caratteristiche:

- molti utenti concorrenti che richiedono i servizi;
- una logica applicativa complessa;
- archivi di grandi dimensioni con organizzazione di dati complessa e distribuita;
- notevoli requisiti di sicurezza;
- sistemi transazionali.

Lo schema di funzionamento di un modello client-server è sintetizzato nella seguente figura:

1. il client manda una richiesta al server;
2. il server (in attesa) riceve la richiesta;
3. il server esegue il servizio richiesto (generando un thread concorrente);
4. il server manda una risposta ed eventualmente dei dati;
5. il client riceve la risposta ed eventualmente i dati.



Il modello client-server

Questo meccanismo è utilizzato sia che i processi fisicamente si trovino in esecuzione su due calcolatori diversi connessi in rete, sia che risiedano sul medesimo calcolatore.

Alcuni servizi tipici delle architetture client-server:

- Telnet: mediante un tale programma (programma client) è possibile operare su un computer remoto come si opera su un computer locale; questo è possibile se sulla macchina remota è presente un programma server in grado di esaudire le richieste del client Telnet;
- HTTP: il browser è un client HTTP (Web), che richiede pagine Web ai computer su cui è installato un Web server, il quale esaudirà le richieste spedendo la pagina desiderata;
- FTP: tramite un client FTP è possibile copiare e cancellare file su un computer remoto, purché qui sia presente un server FTP;
- altri servizi di questo tipo sono SMTP, IMAP4, NFS, NIS e così via.

La programmazione di rete assume sempre più importanza in quanto la maggior parte delle applicazioni ha bi-sogno di reperire o scambiare dati presenti (o recuperabili) su altri PC e quindi la necessità prima è quella della connessione reciproca

Distinzione tra server e client

Un programma chiamato client richiede dei servizi a un altro programma chiamato server. Quest'ultimo è ospitato su un computer chiamato host ed è in ascolto tramite un socket su una determinata porta, in attesa che un client richieda la connessione: il client invia la richiesta al server tentando la connessione proprio tramite tale porta, quella cioè su cui il server è in ascolto.

Un client, quindi, per comunicare con un server usando il protocollo TCP/IP deve, per prima cosa, “connettersi” al socket dell'host dove il server è in esecuzione specificando l'indirizzo IP della macchina e il numero di porta sulla quale il server è in ascolto.

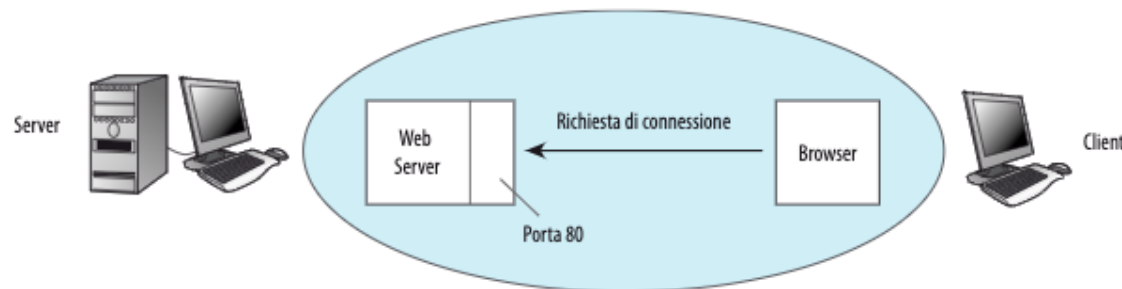
Un socket è formato dalla coppia <indirizzo IP: numero della porta> che permette di individuare univocamente il gestore di un servizio: verrà descritto nella prossima lezione.

Distinzione tra server e client

Naturalmente su uno stesso computer possono essere in esecuzione server diversi, in ascolto su porte diverse (per semplificare con un'analogia, si può pensare al fatto che più persone abitano allo stesso indirizzo, ma a numeri di interno diversi: i numeri di interno rappresentano le porte).

Un server "rimane in ascolto" su una determinata porta finché un client non crea una comunicazione con il socket specificando la porta sulla quale è disponibile il servizio; quindi esegue le richieste del client con le risorse che ha a disposizione e rispedisce, se richiesto, i risultati al client.

La figura seguente riporta un esempio di servizio HTTP, dove il Web server rimane in attesa della connessione dei browser dei client sulla porta 80.



Comunicazione unicast e multicast

Ora che è stato introdotto il concetto client-server, possiamo passare a distinguere due tipi di comunicazione:

- unicast: il server comunica con un solo client alla volta accettando una richiesta di connessione solo se nessun altro client è già connesso;
- multicast: al server possono essere connessi più client contemporaneamente.

Nel caso di trasmissione multicast se la richiesta di connessione tra cliente server va a buon fine il server, prima di stabilire il canale di connessione con il client, sposta la richiesta dalla porta nella quale è stata effettuata, port address, su una nuova porta, così lascia libera la prima in attesa di altre connessioni, e manda in esecuzione un thread che soddisfa la richiesta.

