TFA 2014/15
SISTEMI E RETI DI CALCOLATORI PER L'INSEGNAMENTO
UNITA' DIDATTICA
PROTOCOLLO HTTP

14/05/2015

Laura Ricci

INTERNET: LE APPLICAZIONI

- Applicazioni tradizionali:
  - Email
  - News
  - Remote Login (SSH)
  - File Transfer
- L'applicazione 'killer':
  - World-Wide Web (WWW)
- Nuove applicazioni:
  - Videoconferenza
  - Telefonia (VOIP)
  - P2P applications

APPLICAZIONI DI RETE

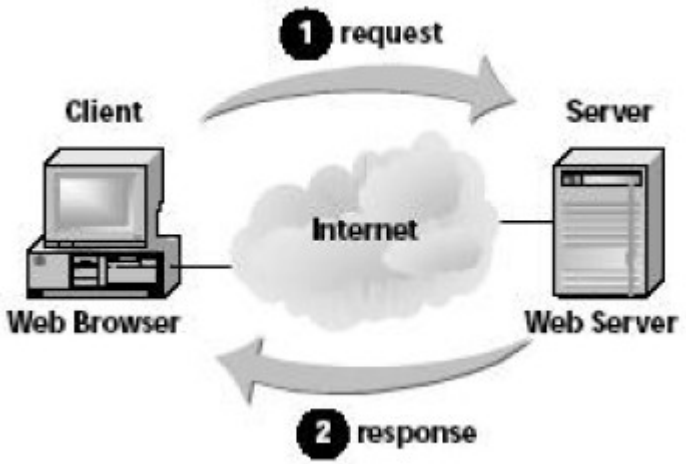
Applicazione	Protocollo a livello applicazioni	Protocollo a livello trasporto
Posta Elettronica	SMTP	TCP
Login Remota	SSH	TCP
Web	HTTP	TCP
Trasferimento File	FTP	TCP
File Server Remoto	NFS	UDP o TCP
Multimedia	Proprietario	UDP o TCP
Telefonia	Proprietario	UDP
DNS (Domain Name Server)	….	…..

IL WORLD WIDE WEB

- World Wide Web (WWW): applicazione sviluppata per l'accesso ad informazioni memorizzate sugli host di una WAN
- WEB: applicazione client/server:

WEB Client = Web Browser programma (Chrome, Explorer, Mozilla, Opera,…) dotato di una interfaccia grafica che consente di reperire e visualizzare informazioni memorizzate su un server web

WEB Server: servizio di gestione pagine web codificate in HTML



WORLD WIDE WEB

- sul web, ogni documento è localizzato mediante una URL (Uniform Resource Locator) o URI (in realtà sono due concetti leggermente diversi, ma semplifichiamo per il momento la presentazione).

- una URL in genere indica un file HTML, ma può indicare anche qualsiasi altro documento accedibile mediante un web browser

- esempio: URL **`http://www.cs.princeton.edu/index.html`** specifica che un documento HTML di nome **`index.htlm`** può essere acceduto mediante HTTP su un host di nome **`www.cs.princeton.edu`**

- una URL è composta delle seguenti parti:
  - nome del protocollo utilizzato per accedere l'oggetto (**`http`**)
  - nome dell'host (**`www.cs.princeton.edu`**)
  - **`pathname`** che individua l'oggetto all'interno dell'host

URL (UNIFORM RESOURCE LOCATOR)

Formato generale di una URL

protocol://hostname :port/path/filename?query#fragment

protocol : file,ftp,http,https,news,telnet,wais,...

hostname : nome simbolico o indirizzo IP del server

port : numero della porta, serve per il demultiplexing,può essere omesso se il server è in esecuzione sulla porta di default (es:la porta 80 per i server HTTP)

path: punta alla directory che contiene la risorsa. Il path è relativo alla document root del server

filename: indica un file specifico nella directory specificata.

query: è una stringa utilizzata per fornire argmenti di forms in esecuzione sul server

fragment: si riferisce ad una parte particolare della risorsa (es: un anchor in un documento HTML

URL (UNIFORM RESOURCE LOCATOR)

- Il document root riferito nel campo path non punta necessariamente alla radice del filesystem del server (infatti il server non necessariamente pubblica il suo intero file system ai clients)
- Esempio: su un server Unix, tutti i file che possono essere acceduti dai clients possono trovarsi nella directory `/var/public/html/`. Questa directory può apparire ai clients come la directory root del server
- Molto spesso viene omesso il filename (ed i campi successivi). In questo caso il server può:
  - inviare un file di default (definito al momento della configurazione del server). Molto spesso il file di default è **`index.html`**
  - inviare una lista dei files e delle sottodirectory presenti nella document root
  - inviare un messaggio di errore **`403 Forbidden error message`**

WORLD WIDE WEB: CARATTERISTICHE

- abbiamo effettuato l'accesso ad una pagina web tramite JAVA, ma come avviene realmente l'interazione tra browser e web server?

- accesso ad un oggetto individuato da una URL, ad esempio

  http://www.cs.princeton.edu/index.html

  - apertura di una connessione TCP dal browser web al server web di nome **http://www.cs.princeton.edu** (porta 80 di default)

  - invio di un messsaggio di richiesta sulla connessione aperta utilizzando il protocollo HTTP.

  - il server web ricerca l'oggetto individuato dal path **index.html** ed invia il contenuto del file al browser

  - il server chiude la connessione con il client (connessione non persistente)

WORLD WIDE WEB: CARATTERISTICHE

- Hypertext Transfer Protocol (HTTP): protocollo di livello applicazione di tipo richiesta/risposta utilizzato per l'invio di contenuto web

- Ogni pagina web visitata, ad esempio una pagina Facebook, è scaricata utilizzando HHTP

- HTTP è stato esteso rispetto all'originale progettato solo per il trasferimento di pagine web ed è attualmente utilizzato come base per altri protocolli, esempio il protocollo SIP, utilizzato per VOIP

WORLD WIDE WEB: PROTOCOLLO HTTP

Protocollo di Rete: definisce le regole e le convenzioni per lo scambio di messaggi tra due entità di una rete

Protocollo HTTP: insieme di regole che definiscono formati e ordine dei messaggi scambiati tra un browser ed un server web.

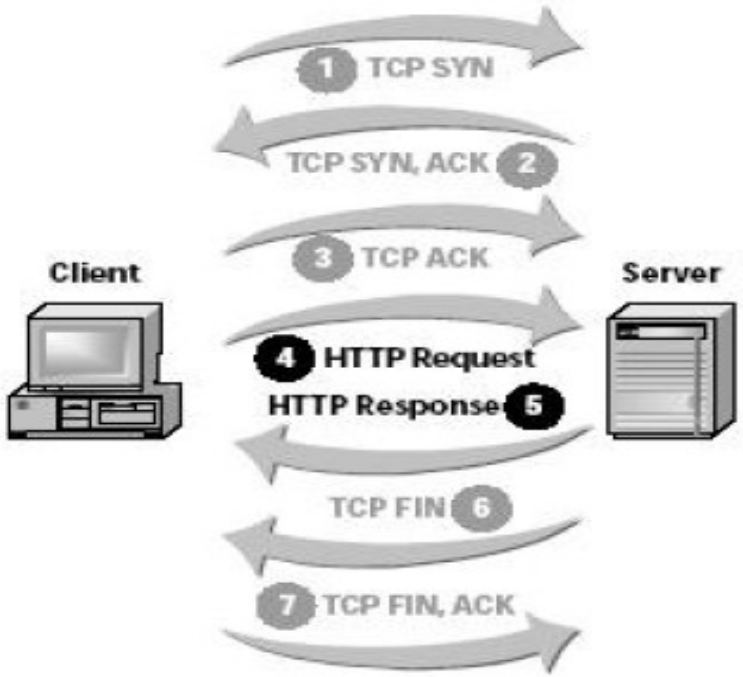
- protocollo a livello applicazione
- utilizza il livello di trasporto TCP
- protocollo privo di stato:
  - non memorizza informazioni relative alle richieste ricevute dagli utenti (eccezione: vedere meccanismo cookies)
  - ogni richiesta da parte di un utente viene trattata in modo indipendente, non si mantiene memoria delle richieste fatte in precedenza dal solito utente
  - non esiste una nozione di sessione, come nel protocollo Telnet o FTP
  - ma....diversi meccanismi introdotti successivamente per introdurre una qualche forma di stato

HTTP: PROTOCOLLO STATELESS

- da notare che il server invia gli oggetti richiesti ai client senza immagazzinare alcuna informazione di stato relativa al client.
- se uno stesso client chiede lo stesso oggetto due volte nel giro di pochi secondi, il server lo rispedisce come se avesse completamente dimenticato ciò che ha appena fatto
- poichè un server HTTP non conserva le informazioni relative ai client, l'HTTP è detto **protocollo senza stato (*stateless protocol*).**

HTTP: CONNESSIONI NON PERSISTENTI

HTTP 1.0 connessioni non persistenti: viene aperta una nuova connessione per ogni richiesta effettuata dal client al server. La connessione si chiude con l'invio della risposta da parte del server



HTTP: CONNESSIONI NON PERSISTENTI

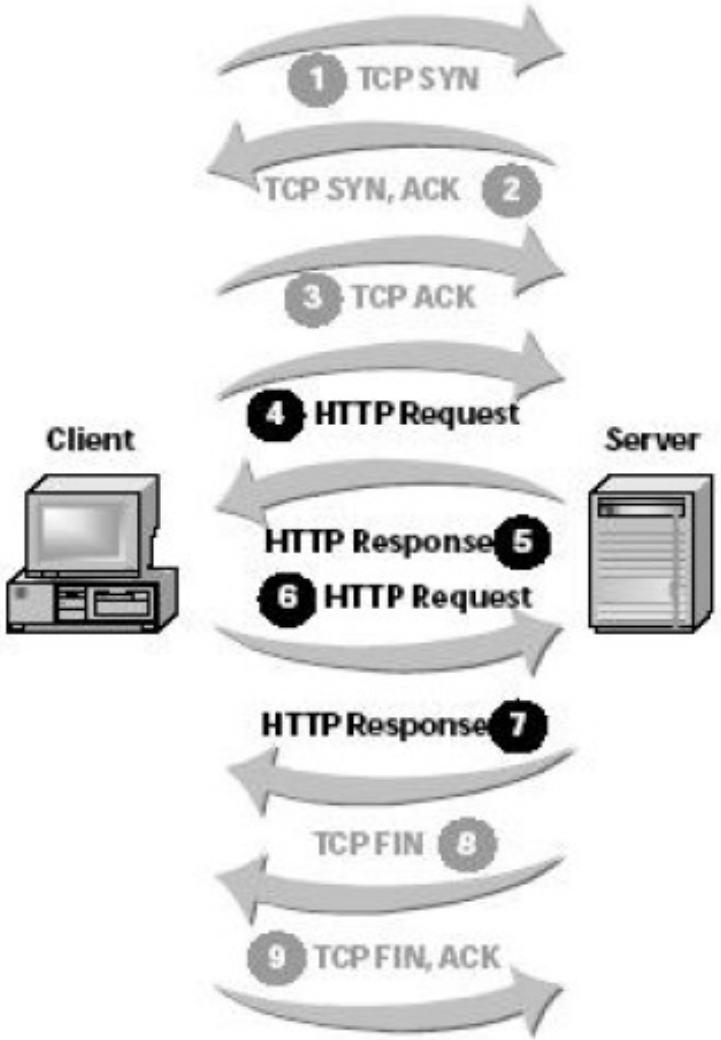
Connessioni non persistenti, svantaggi: supponiamo che un client richieda una pagina WEB ad un server web. La pagina contiene un file di base HTML e 10 immagini JPEG. Tutti gli 11 oggetti risisedono sullo stessp server.

- il server invia il file HTML di base al client e subito dopo chiude la connessione TCP
- il client(browser) riceve la pagina HTML e vede che in essa sono contenuti 10 oggetti JPEG. Ogni oggetto è identificato da una URL
- vengono stabilite, in sequenza, 10 connessioni con il server per il reperimento delle 10 immagini
- nei browser moderni esiste la possibilità di gestire più connessioni TCP contemporaneamente

PROTOCOLLO HTTP: CONNESSIONI PERSISTENTI

- svantaggi HTTP 1.0:
  - l'apertura ripetuta di connessioni TCP comporta l'allocazione/deallocazione di buffers per la gestione della connessione
  - 3 way handshake ripetuto per ogni connessione aparta
- HTTP 1.1 utilizza connessioni persistenti:
  - più messaggi di richiesta/risposta sulla stessa connessione TCP
  - la connessione rimane aperta anche dopo che una richiesta del client è stata servita
  - il server può inviare sulla stessa connessione tutte le risorse che compongono una pagina HTML
  - la stessa connessione può essere utilizzata per l'invio di pagine HTML diverse
  - la connessione viene chiusa quando rimane inattiva per un certo intervallo di tempo (uso di timeout)

HTTP: CONNESSIONI PERSISTENTI



HTTP: CONNESSIONI PERSISTENTI

Connessioni persistenti (HTTP 1.1): il server lascia aperta la connessione TCP dopo aver inviato la risposta al client, in attesa di ulteriori richieste

Vantaggi

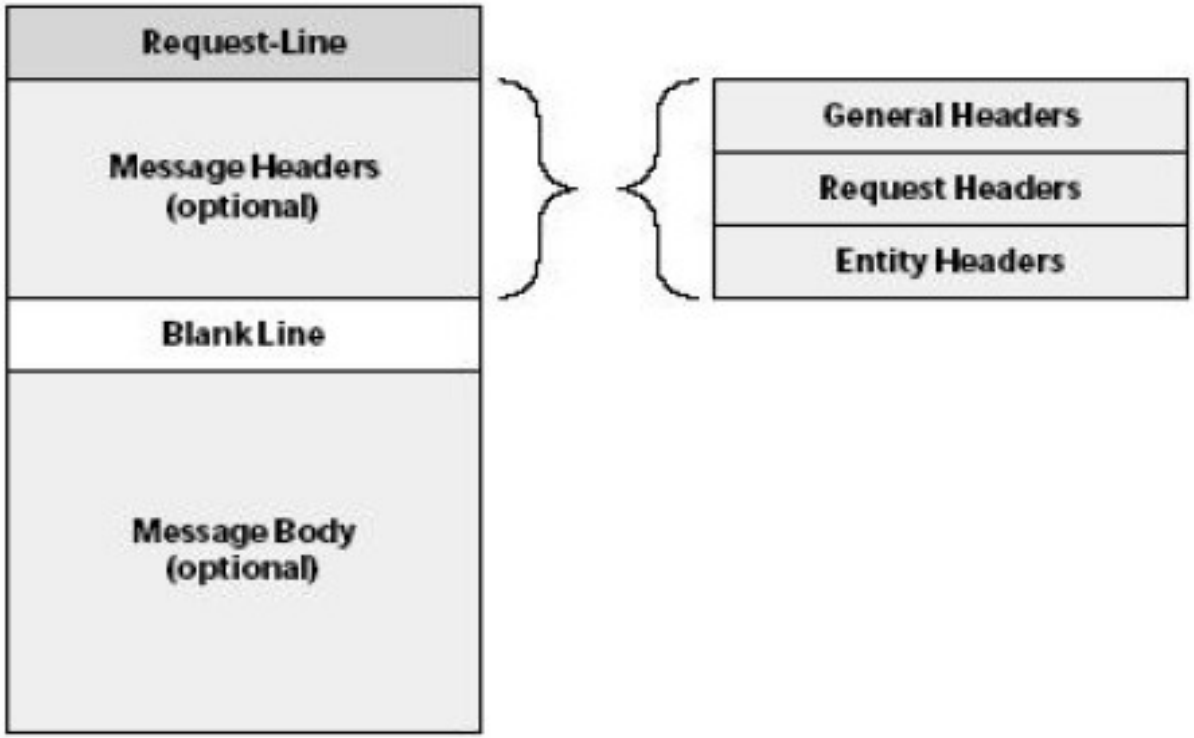
- minor overhead nella creazione di connessioni
- miglior utilizzazione del meccanismo TCP di controllo di congestione

Problemi

nel server: decidere il momento in cui chiudere una connessione

uso di timeouts = il server chiude una connessione dopo che è trascorso un intervallo di tempo in cui non ha ricevuto richieste dal client

HTTP: STRUTTURA DEI MESSAGGI



HTTP: LINEA DI RICHIESTA

E' una linea di testo è composta da tre campi, separati da spazi:

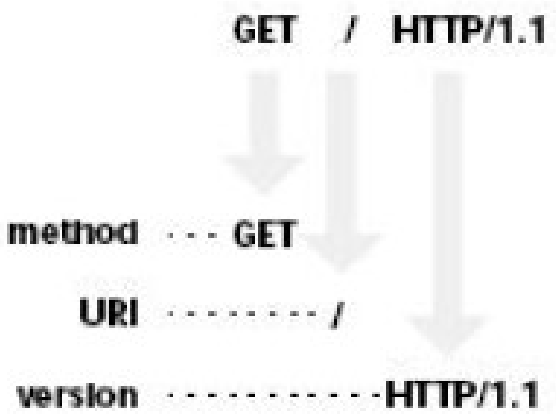
- metodo
- URI
- versione HTTP

essa indica quindi

- il metodo che il client sta richiedendo
- la risorsa alla quale applicare il metodo
- la versione HTTP che si sta usando.

Esempio di linea di richiesta

GET / HTTP/ 1.1



RICHIESTA CON HEADER

Per trovare il file all'URL:

http://www.somehost.com/path/file.html

- si apre una connessione con l'host **www.somehost.com**, porta 80 (si usa la porta di default perchè non ne è specificata una nell'URL).
- si manda un messaggio di richiesta tipo il seguente al socket stesso

```
GET /path/file.html HTTP/1.1
Host: www.somehost.com
Connection: close
User-Agent: Mozilla/4.0
Accept-Language: En-us
Accept-Encoding: gzip, deflate(ritorno carrello extra, line feed)
```

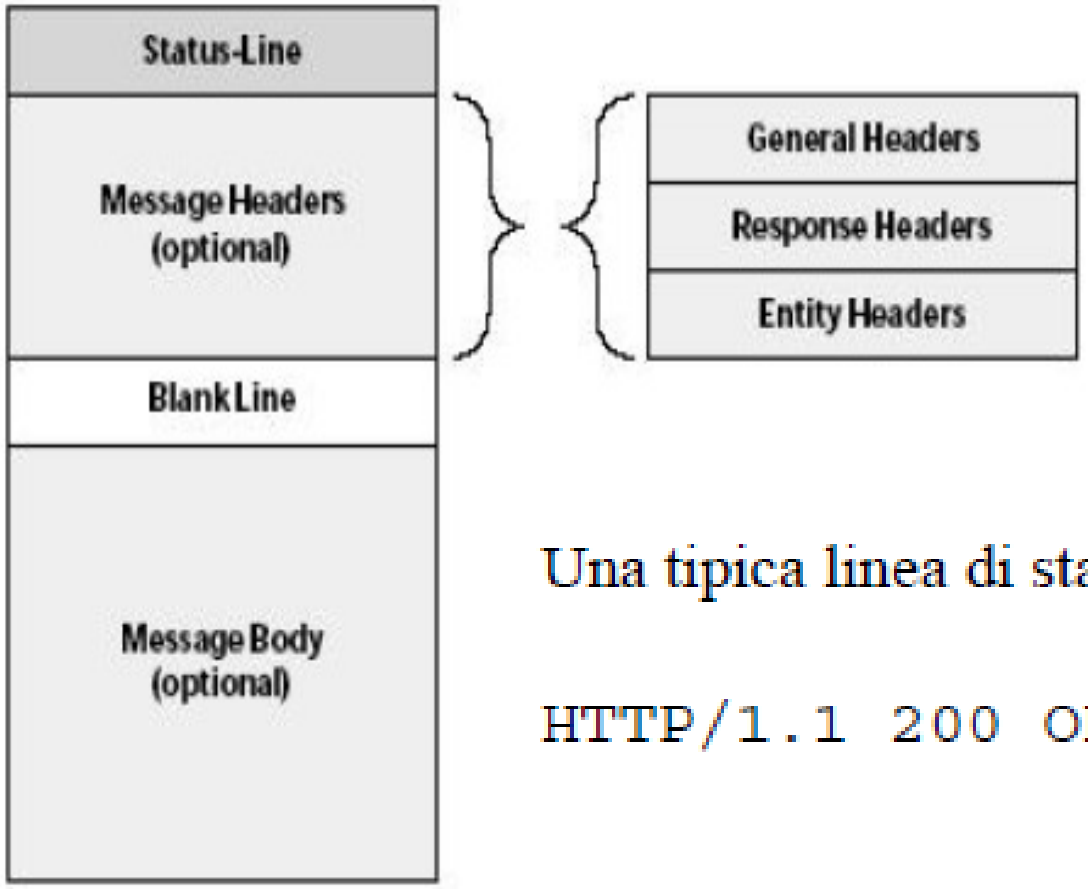
HTTP: RISPOSTA DEL SERVER

- Dopo che un server Web riceve una richiesta HTTP, intraprende le azioni necessarie a fornire la risorsa richiesta.
- queste possono includere l'esecuzione di uno script o altro.
- il server Web risponde al client con una risposta HTTP, organizzata in modo simile ad una richiesta HTTP.
- L'unica differenza significativa è che le risposte cominciano con una linea di stato piuttosto che con una linea di richiesta.

Quindi, come nel caso di una richiesta, una risposta HTTP comprende tre elementi di base:

- linea di stato
- intestazioni HTTP
- corpo del messaggio

LINEA DI STATO NELLA RISPOSTA



Una tipica linea di stato è:

```
HTTP/1.1 200 OK
```

METODO GET

- la più semplice operazione HTTP è GET.
- serve a un client per recuperare un oggetto dal server.
- sul Web, un browser richiede una pagina da un server con un GET. Questo è

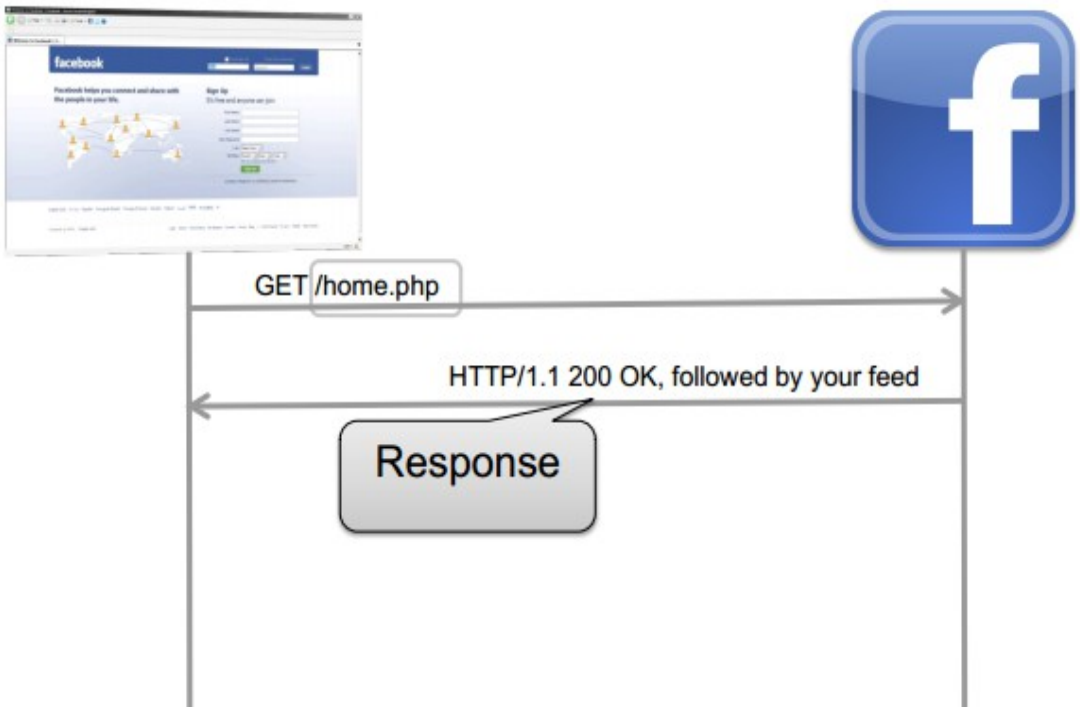
il tipo di richiesta che un browser usa quando l'utente clicca su un link o immette un URI nell'apposito campo del browser.

HTTP PER ACCESSO A FACEBOOK

Ogni richiesta include tre parti:

- azione (metodo) + risorsa sul server
- una serie di header che contengono metadati
- un messaggio (opzionale)

HTTP PER ACCESSO A FACEBOOK

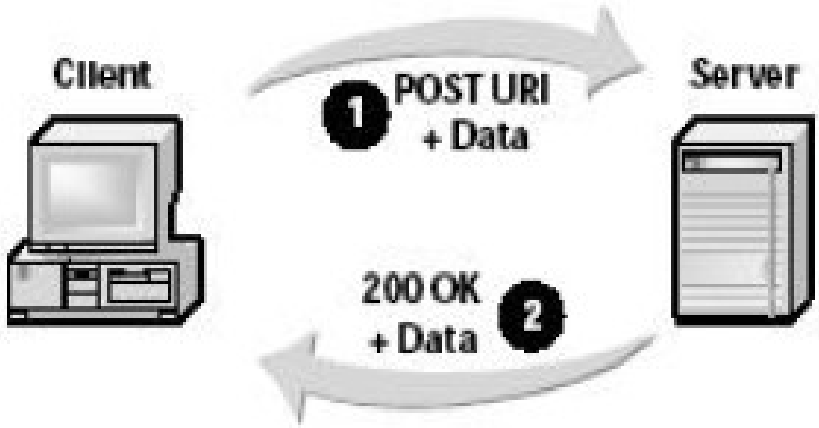


Codici di stato comuni:

- `200 OK` : è andato tutto bene
- `404 Not Found` : la risorsa non è stata trovata
- `500 Server Error` : errore in uno script sul server
- `301 Moved Permanently` : la risorsa è stata spostata altrove

PROTOCOLLO HTTP: POST

- a differenza di GET, che permette a un server di mandare informazioni a un client, l'operazione POST fornisce ai client un modo per trasmettere informazioni ai server.
- la maggior parte dei browser web usa comunemente tale metodo per inviare forms ai server Web.
- il client manda un messaggio POST e include l'informazione che desidera mandare al server.



PROTOCOLLO HTTP: POST

- anche in un metodo POST viene specificata una URI.
- in questo caso la URI identifica l'oggetto nel server che può elaborare

l'informazione inclusa.

- sui server Web, questo URI è frequentemente un programma o uno script.
- un server può rimandare l'informazione stessa come parte della risposta.
- tipicamente, essa è una nuova pagina da visualizzare, spesso influenzata dai

dati inseriti dall'utente; ad esempio, nel caso di una form di ricerca, come

avviene in un motore di ricerca, la nuova pagina Web spesso mostra i

risultati della richiesta.

- l'URI richiesto non è una risorsa da prelevare; è di solito un programma per

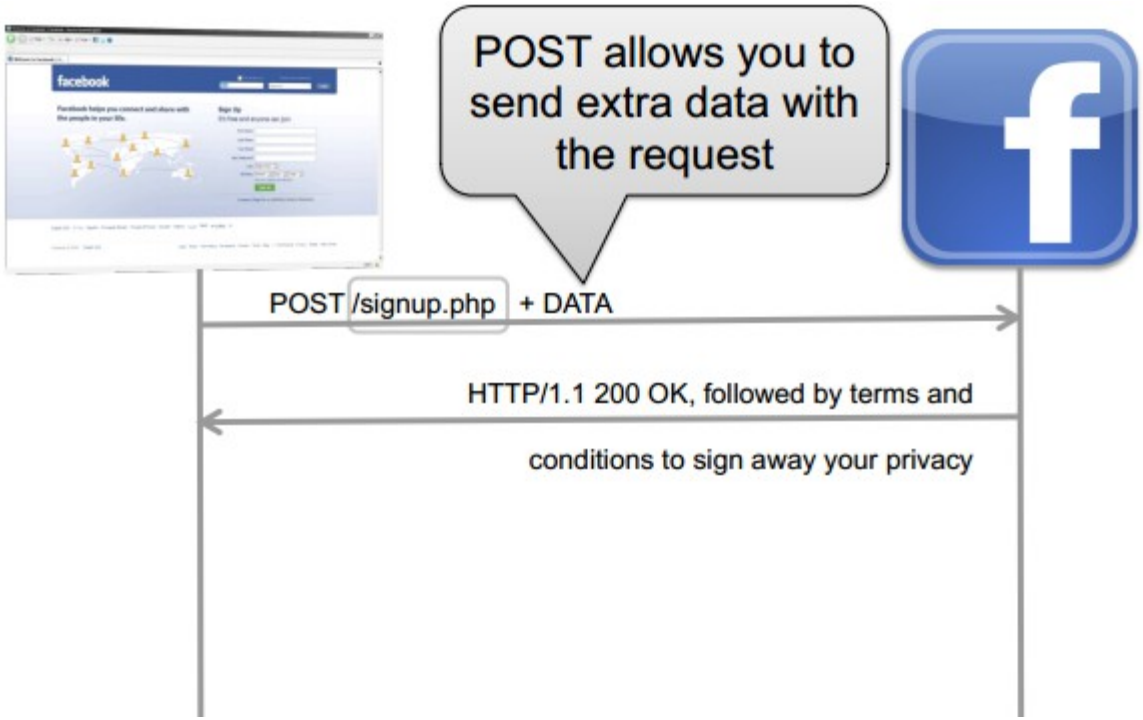
maneggiare i dati che si stanno spedendo.

- la risposta HTTP è normalmente l'uscita di un programma, non un filestatico.

PROTOCOLLO HTTP IN BREVE

Occorre un metodo che consenta al client di inviare un messaggio al server

HTTP POST



Metodo POST: una richiesta POST include, oltre alla risorsa da accedere, ulteriori parametri parametri: serie di coppie chiave-valore
Ad esempio:

- `FirstName=John`
- `LastName=Doe`
- `SignAwayPrivacy=True`

HTTP POST: UN ESEMPIO

```
POST /login.php?login_attempt=1 HTTP/1.1
Accept:application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Content-Type:application/x-www-form-urlencoded
Origin:http://www.facebook.com
Referer:http://www.facebook.com/
User-Agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_3; en-US) AppleWebKit/533.4 (KHTML, like Gecko)
Chrome/5.0.375.53 Safari/533.4
charset_test:€,´,€,´,ж,Д,€
locale:en_US
email:john.doe@gmail.com
pass:somepassword
```

PROTOCOLLO HTTP: MAGGIORI DETTAGLI

- Messaggio codificato in ASCII
- Ogni messaggio è composto da alcune righe separate da un CR (carriage return=invio), LF(Line Feed=nuova linea)
- La prima riga viene detta riga di richiesta, le successive righe di intestazione (headers), segue (eventualmente) il corpo del messaggio
- Riga di richiesta include
  - campo metodo (`GET`, `POST`, `HEAD`)
  - campo URL
  - campo versione
- Ogni riga di intestazione include
  - nome del campo di intestazione (es: connection)
  - valore corrsipondente (es: close/keep-alive)

PROTOCOLLO HTTP: MAGGIORI DETTAGLI

- HTTP headers: specificano metadati inviati dal client al server per "aiutare" il server ad elaborare correttamente la richiesta
- coppie: (Nome, Valore)
- alcuni esempi di header importanti: specificano il tipo di risposte che un client è disposta ad accettare da un server
  - **`Accept`**: i tipi di contenuti che il client può accettare
  - **`Accept-Charset`**: il tipo di caratteri che il client può gestire
  - **`Accept-Language`**: non tutti accettano l'inglese

PROTOCOLLO HTTP: MAGGIORI DETTAGLI

- HTTP headers: specificano metadati inviati dal client al server per "aiutare" il server ad elaborare correttamente la richiesta

- un altro header: User Agent header specifica il tipo del client che ha fatto la richiesta (e.g. Firefox + version, Android + version, etc.)

- Perchè il server è interessato a questa informazione?
  - Il server modifica la presentazione del contenuto a seconda dell'user agent
  - Esempio: presentazione diversa dei contenuti per device mobili

UN ESEMPIO

- Esempio di interazione HTTP: supponiamo che un utente abbia inserito la URL http://www.di.unipi.it/~ricci/papers.html
- Dopo che la URL è stata inserita, il web browser attiva un client HTTP che stabilisce una connessione sulla porta 80 del server HTTP dell'host

  www.di.unipi.it
- Formato del messaggio HTTP di richiesta inviato dal client al server

```
GET ~ricci/papers.html HTTP/1.1 <CR LF>
Accept Language:it <CRLF>
User Agent: Mozilla/4.0 <CRLF>
Host:  131.114.3.18 <CRLF>
Connection: keep alive <CRLF>
<CRLF><CRLF>
```

UN ESEMPIO

Risposta del Server:

```
HTTP/1.1 202 OK <CRLF>Date:…. <CRLF>
    Server: Apache/1.3.0 (Unix) <CRLF>
  Connection: close <CRLF>
  Last Modified: Mon 24 Oct 2007<CRLF>
  Content Length: 6821<CRLF>
  Content Type: text/HTML<CRLF><CRLF>
  <HTML>
     <BODY>
    <H1> LAURA RICCI PAPERS </H1>.............
    </BODY>
     </HTML>
  <CRLF>
```

WORLD WIDE WEB: CARATTERISTICHE

- l'oggetto O inviato dal server S al client (index.html) può contenere collegamenti ipertestuali, cioè URL che puntano ad altri file, allocati su S stesso o su altri servers

- quando l'utente seleziona un collegamento ipertestuale, il browser apre una nuova connessione TCP per recuperare l'oggetto individuato

- se si usano connessioni non persistenti, viene creata una nuova connessione anche nel caso in cui gli oggetti riferiti siano memorizzati su S stesso

LABORATORIO: IL PROTOCOLLO HTTP

- Obiettivo: fare esperienza su protocollo HTTP

- Richiesta di accesso ad una pagina web, stampa del codice HTML restituito da HTTP

- Uso della classe JAVA `HttpURLConnection`

LABORATORIO: IL PROTOCOLLO HTTP

```
import java.net.*; import java.io.*;
public class HttpConnection {
public static void main(String [] args)  throws Exception
     {  URL obj;String s;
        obj = new URL("http://www.di.unipi.it");
        HttpURLConnection con = (HttpURLConnection) obj.openConnection();
        con.setRequestMethod("GET");
        int responseCode = con.getResponseCode();
        System.out.println(responseCode);
        InputStream is= con.getInputStream();
        BufferedReader d
            = new BufferedReader(new InputStreamReader(is));
        while ((s = d.readLine( )) != null) {
            System.out.println(s); } } }
```

WORLD WIDE WEB

- L'esempio fatto semplifica parecchio le cose
- Nella realtà occorre tenere in considerazione
  - Web caching
  - Pagine dinamiche
  - Esecuzione remota di programmi sul server
  - Applets
  - Sicurezza
  - Cookies

AUTENTICAZIONE DEI CLIENTS

Meccanismo soft di autenticazione mediante username + password

client: invia la richiesta

server: risponde con un messaggio di richiesta autenticazione

client: visualizza la richiesta di autenticazione ed attende i dati dall'utente. Rispedisce al server username+password e le memorizza (caching)

server: autentica l'utente

client: rispedisce username+password in tutte le richieste successive utilizzando quelle memorizzate nella cache

COOKIES: MOTIVAZIONI

- HTTP è un protocollo stateless
- invocazioni successive di funzioni HTTP costituiscono eventi indipendenti, e non vi è alcun meccanismo di interazione diretta
- senza il concetto di stato, è difficile per il server

  tenere traccia delle richieste successive di un client

  creare applicazioni web che "ricordino" cosa è accaduto nelle interazioni precedenti

  associare ad ogni client una identità unica.

- Per ovviare a questi svantaggi si introduce meccanismo dei cookies

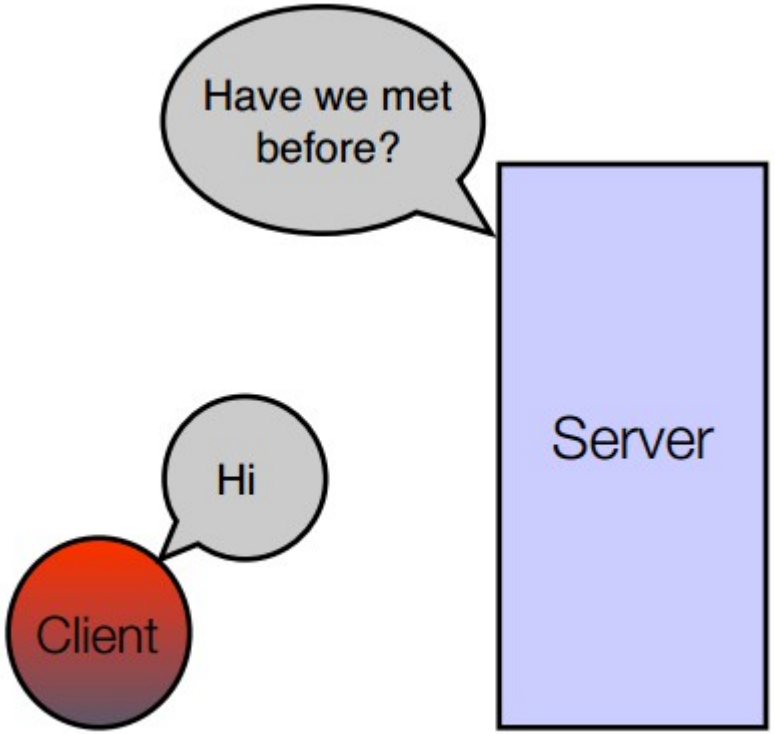
IL MECCANISMO DEI COOKIES

- HTTP cookies (web cookies, cookies) : uno dei modi per introdurre il concetto di "stato" nel protocollo HTTP

- motivazione per i cookie

- come funzionano in HTTP

- generazione di identificatori ur clients o sessioni



IL MECCANISMO DEI COOKIES

- HTTP cookies (web cookies, cookies) : uno dei modi per introdurre il concetto di "stato" nel protocollo HTTP
- cookie: una sequenza di dati in formato testuale (un piccolo file) spediti dal server al client e, successivamente, rispediti dal client al server
  - spesso (ma non sempre) il dato è una stringa che rappresenta un identificatore unico per il client
- sono dati (identificatori) generati dal server, non sono delle porzione di codice

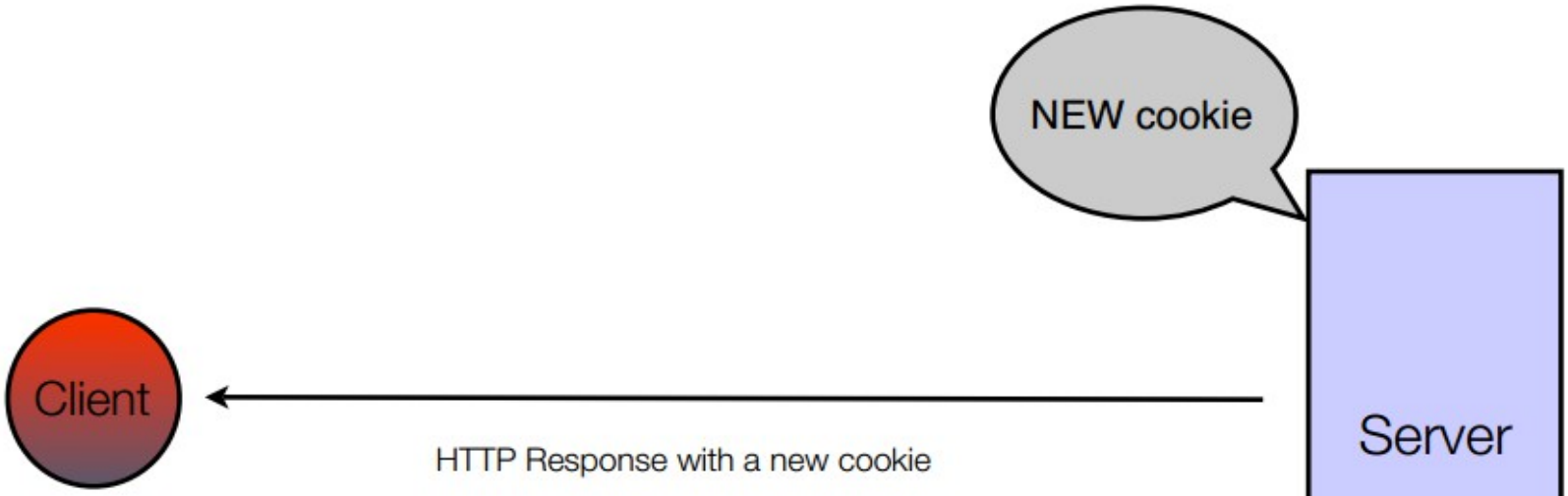
un cookie viene:

- generato dal server quando un client, tramite un browser, effettua una richiesta HTTP
- inviate da un web server al browser (client)
- rispedite senza modifiche dal browser al web server ogni volta che il browser accede nuovamente al sito

UN SEMPLICE SCENARIO: PASSO 1

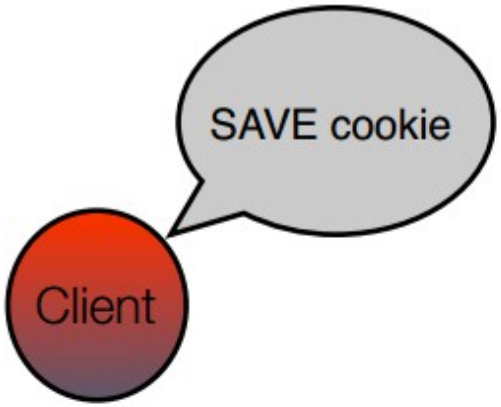
The client (typically a web browser) makes a request to the server for the first time.

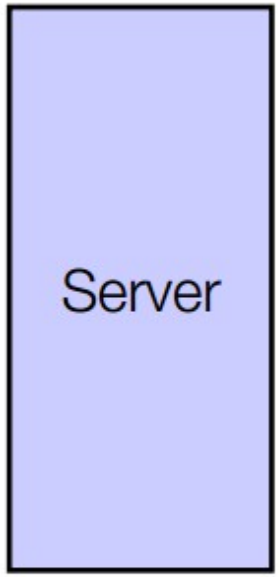
UN SEMPLICE SCENARIO: PASSO 2



The server observes that the client has not sent a cookie. So the server generates a new unique cookie and sends it back to the client as part of its response.

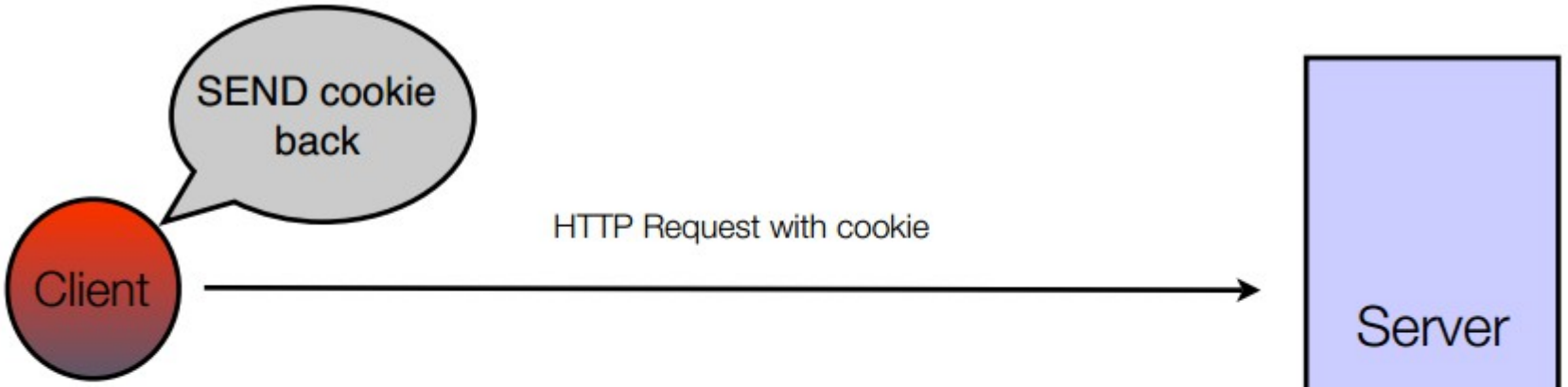
UN SEMPLICE SCENARIO: PASSO 3





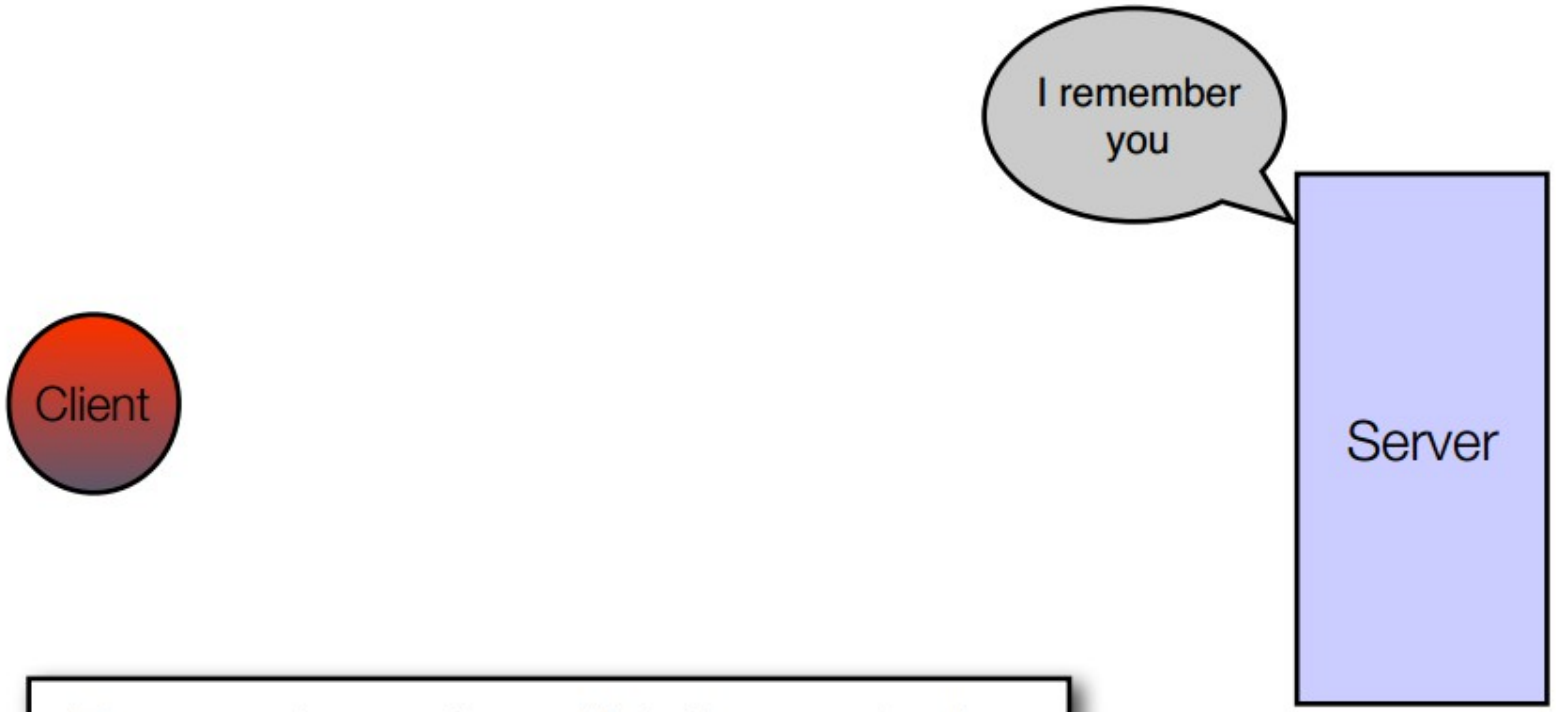
The client saves the new cookie from the server in its private store of cookies. It associates the cookie with the domain name of the server.

UN SEMPLICE SCENARIO: PASSO 4



Later, the client sends another request to the server. This time it observes that is has a cookie saved for the domain name of the server, so it sends the cookie as part of its request.

UN SEMPLICE SCENARIO: PASSO 5



The server observes the cookie in the request and remembers that it sent the cookie in a previous response. The cookie allows the server to identify the client.

IL MECCANISMO DEI COOKIES

- La maggior parte dei browser moderni supportano il meccanismo dei cookies, ma danno all'utente la possibilità di scegliere se accettare i cookie o meno

- In particolare, il browser può

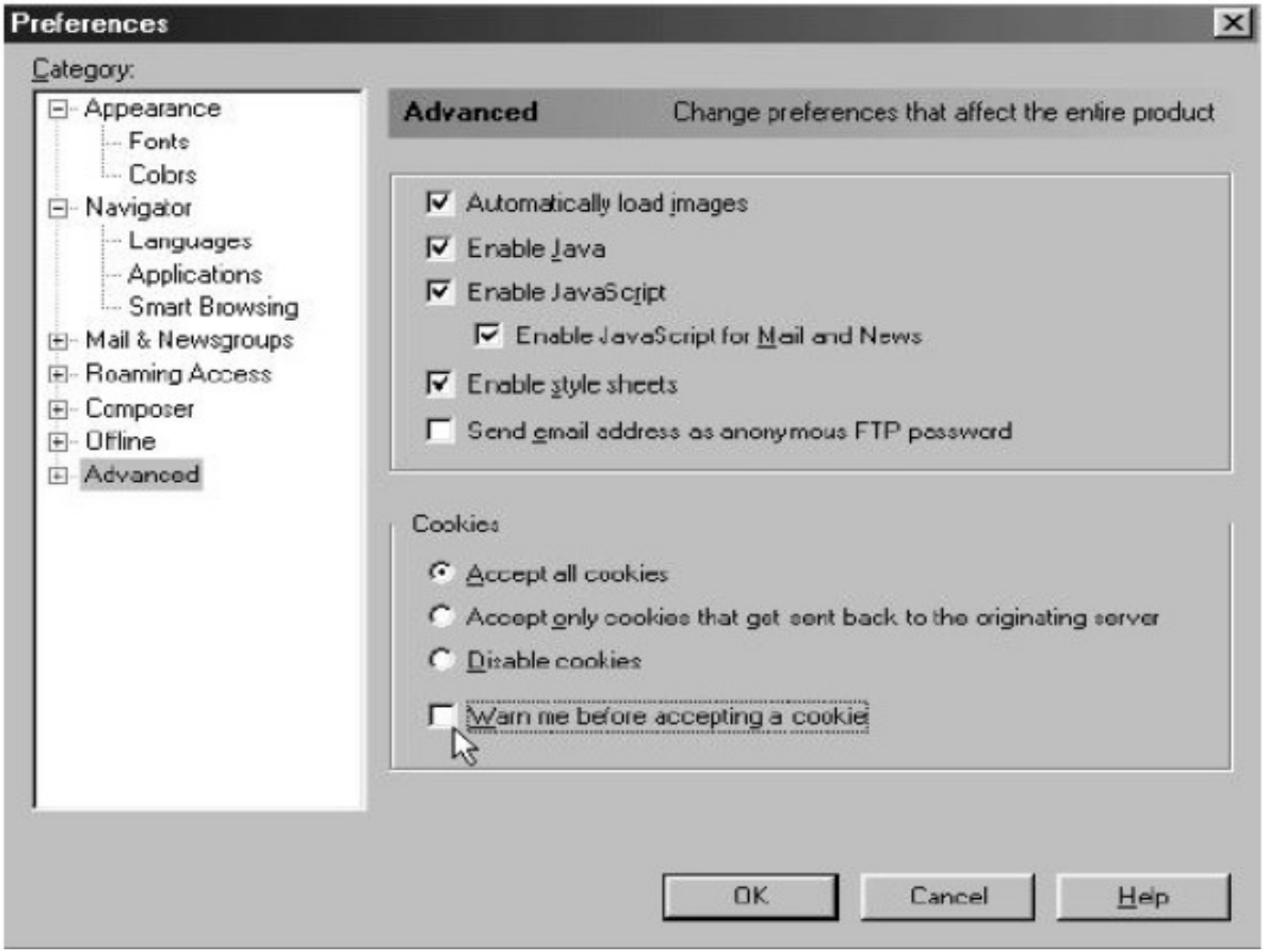
  non accettare mai i cookie

  chiedere all'utente se accettare o meno il cookie

  accettare sempre i cookie

  accettare un cookie solo se contenuto in una lista fornita dall'utente

ACCETTAZIONE DI COOKIES



COOKIES: UTILIZZO

- senza i cookie ogni accesso ad una pagina web è un evento isolato (stateless HTTP transaction)

- restituendo un cookie al server, il browser fornisce al server un modo di collegare la visita corrente alle visite precedenti dello stesso client allo stesso sito

- possibili utilizzi:
  - ricordare le pagine visitate precedentemente e proporre all'utente dei contenuti personalizzati
  - autenticare un utente senza richiedere username+password ogni volta che l'utente si collega al server
  - ricordare le preferenze dell'utente (es: pagine visitate,…) per riproporle successivamente

COOKIES: UTILIZZO

- ricordare le preferenze dell'utente (es: pagine visitate,…) per riproporle successivamente.

- l'utente seleziona le sue pagine preferite ed il server crea un cookie che individua le sue preferenze. Al momento dell'accesso successivo, si mostra una pagina personalizzata.

- autenticare un utente senza richiedere username+password ogni volta che l'utente si collega al server. Accesso aperto agli utenti precedentemente registrati

- gestione di un carrello elettronico per un sito di commercio elettronico: memorizzazione nel cookie dei prodotto acquistati
  - il server può calcolare il costo totale in base al cookie

LABORATORIO: ACCEDERE MEDIANTE URL

- Prerequisito: conoscenza (anche minima) di HTML
- Scopo del laboratorio: fare esperienza pratica su URL: una URL che individua un file HTML che rappresenta una pagina web
- Scrivere un programma JAVA che consenta di accedere ad un file HTML specificando la URL che lo individua
- Scaricare il file HTML, salvarlo in un file e rileggere il file mediante un browser, poi confrontare il file aperto dal browser con la pagina acceduta in remoto

LABORATORIO: ACCEDERE MEDIANTE URL

```
import java.io.*; import java.net.*;
public class URLuse {
    public static void main (String[] args) {
      URL u; InputStream is = null; DataInputStream dis;
      String s;
        try { u = new URL("http://www.di.unipi.it");
            is = u.openStream();
            BufferedReader d
              = new BufferedReader(new InputStreamReader(is));
            while ((s = d.readLine()) != null)
                  {System.out.println(s); } }
      catch (MalformedURLException mue) {
               System.out.println("Ouch - a MalformedURLException happened.");
               System.exit(1); }
      catch (IOException ioe) {
               System.out.println("Oops- an IOException happened.");
                ioe.printStackTrace(); System.exit(1);}}}
```

CACHING

- il caching è uno dei modi più comuni per migliorare le performance dell'HTTP

e, specialmente in Internet, è anche uno dei più efficienti.

- le specifiche HTTP riconoscono l'importanza del caching ed utilizzano a tale

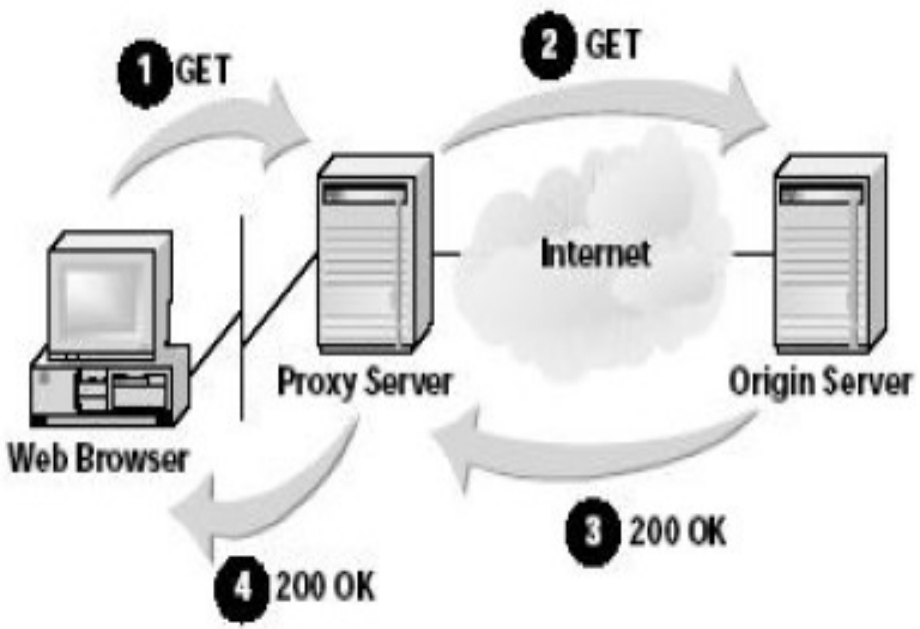
tecnologia estesamente all'interno del protocollo stesso.

- scopo principale di una cache: immagazzinare una copia di un oggetto per prevenire la necessità di recuperarla nuovamente in seguito.

- tre tipi principali di caching:
  - caching tramite proxy server
  - caching sul server
  - caching sul client

PROXY SERVER

- Un proxy server è un'entità della rete che soddisfa le richieste HTTP da parte di un client. Esso è una cache con un proprio disco di archiviazione e conserva nella sua memoria copie degli oggetti richiesti di recente.



PROXY SERVER

- un proxy server compie principalmente quattro operazioni: riceve richieste dai client risponde alle richieste prendendo le risposte dalla sua cache, se possibile

- preleva i documenti da altri server se non sono in cache gestisce la cache dei documenti

- un proxy server determina se deve servire una richiesta o passarla al server di origine in base a regole specificate dall'amministratore del server che specificano,ad esempio, quali documenti devono essere cachati e per quanto tempo.

- l'output di ogni scriptè un documento dinamico che può essere diverso ogni volta che è richiesto, così gli script non possono essere cachati e devono essere richiamati ogni volta che sono richiesti da un client, a differenza dei documenti statici che possono risiedere sulla cache.