BD 2 - Applicazioni Web

Luca Cosmo

Università Ca' Foscari Venezia

Università
Ca'Foscari
Venezia

Introduzione

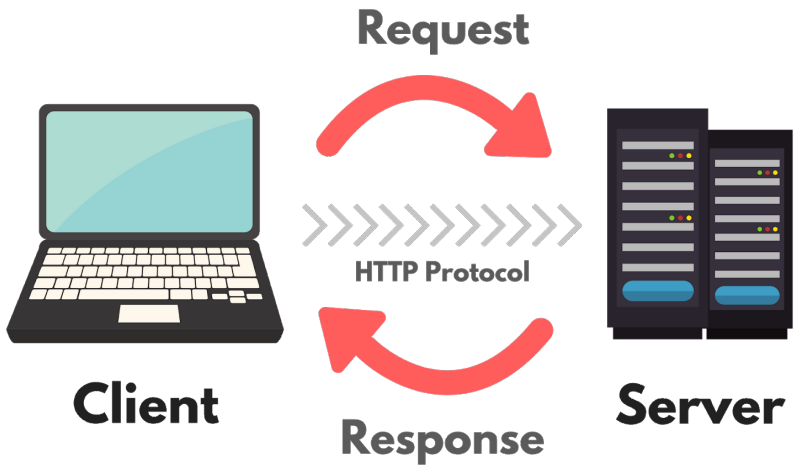
Molte applicazioni che hanno necessità di interfacciarsi con una base di dati nascono in ambito Web:

- e-commerce: per la gestione dei prodotti, degli ordini, etc.
- social network: per la gestione di post, amici, etc.
- streaming: per il catalogo dei film, la watch list, etc.

Il progetto del corso prevede lo sviluppo di una web application e richiede una conoscenza di base di come funziona questo tipo di applicazioni.

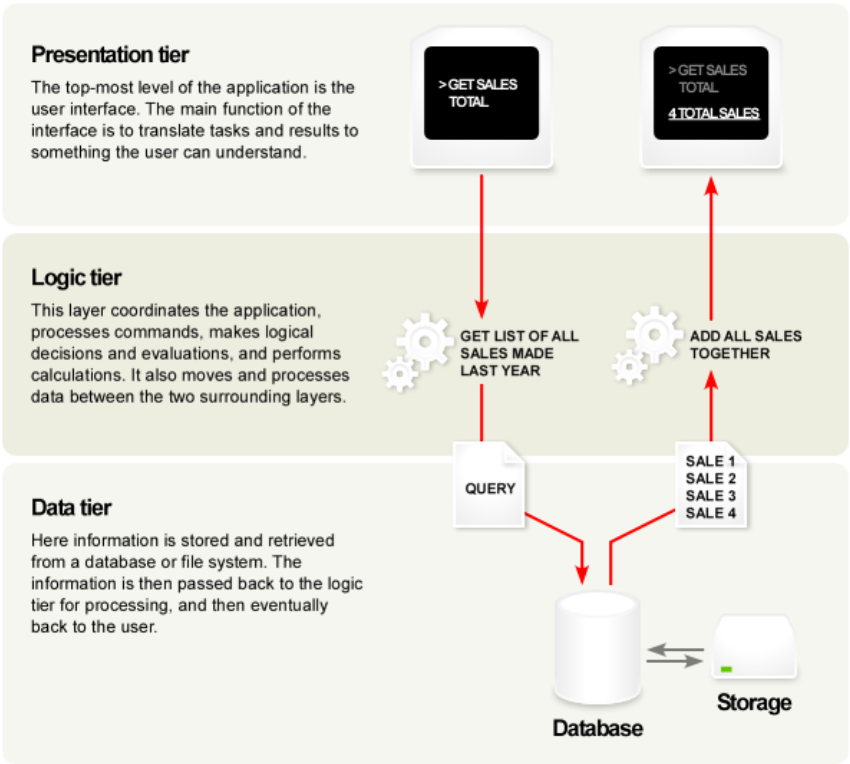
Hyper Text Transfer Protocol (HTTP)

HTTP costituisce le fondamenta del Web: è un semplice protocollo che consente ad un client di richiedere risorse presenti su un server.



Nota: HTTP è un protocollo senza stato per motivi di scalabilità e non fornisce alcuna forma di sicurezza.

Architettura di una Web Application



Tecnologie coinvolte:

- Presentation tier: HTML + CSS + JS
- Logic tier: PHP, JSP, Flask...
- Data tier: MySQL, Postgres, ...

Separazione fra client (presentation) e server (logic + data)

Uniform Resource Locators (URL)

Una risorsa web è identificata da un URL, la cui sintassi (semplificata) è:

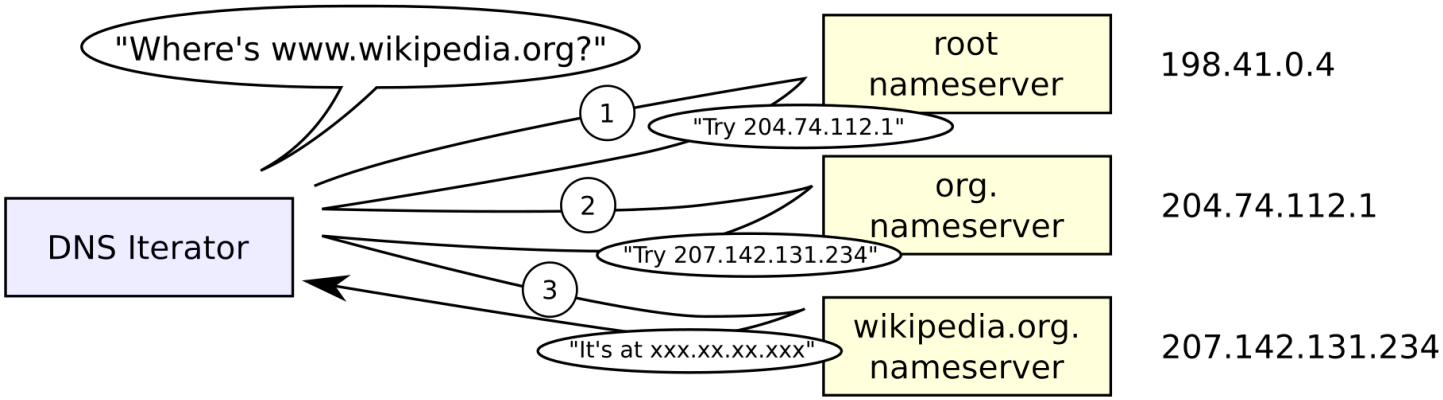
```
schema://hostname[:port]/path[?query][#fragment]
```

dove:

- `schema` è normalmente uno fra `http` e `https`
- `hostname` identifica il server che ospita la risorsa identificata, per esempio `www.facebook.com`
- `port` è un numero di porta (80 per HTTP, 443 per HTTPS)
- `path` identifica la risorsa all'interno del server al punto precedente, per esempio `/users/luca`
- `query` lega parametri a valori (es. `p1=v1&p2=v2...`)
- `fragment` normalmente identifica una parte della pagina HTML

Domain Name System (DNS)

Il protocollo DNS è usato per tradurre un nome di dominio in un indirizzo IP, secondo una relazione molti-a-molti.



Il risultato della traduzione è salvato in una cache per una certa durata.

Richieste HTTP

Le richieste HTTP sono strutturate come segue:

1. una request line, che include un metodo, una risorsa richiesta e la versione del protocollo
2. una lista di request headers, che includa almeno l'header `Host`
3. una linea vuota come separatore
4. un corpo opzionale

Example

```
POST /cart/add.php HTTP/1.1
Host: www.amazon.com

item=56214&quantity=1
```

Metodi HTTP

Lista dei metodi più comuni disponibili in HTTP:

- GET: legge informazione dal server
- HEAD: come GET, ma chiede solo gli header della risposta
- POST: invia informazione al server
- PUT: carica un file sul server
- DELETE: rimuove un file dal server
- OPTIONS: richiede la lista dei metodi supportati

Metodi HTTP

Method	Req. body	Resp. body	Safe	Idempotent
GET	optional	yes	yes	yes
HEAD	optional	no	yes	yes
POST	yes	yes	no	no
PUT	yes	yes	no	yes
DELETE	optional	yes	no	yes
OPTIONS	optional	yes	yes	yes

E' compito del programmatore rispettare questa tabella!

Risposte HTTP

Le risposte HTTP sono strutturate come segue:

1. una status line, che include uno status code ed un messaggio
2. una lista di response headers
3. una linea vuota come separatore
4. un corpo opzionale (HTML nel caso di pagine web)

Example

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<html><body>Done!</body></html>
```

Status Code

Code	Category	Example
2XX	Success	200 OK
3XX	Redirection	301 Moved Permanently
4XX	Client error	401 Unauthorized
5XX	Server error	503 Service Unavailable

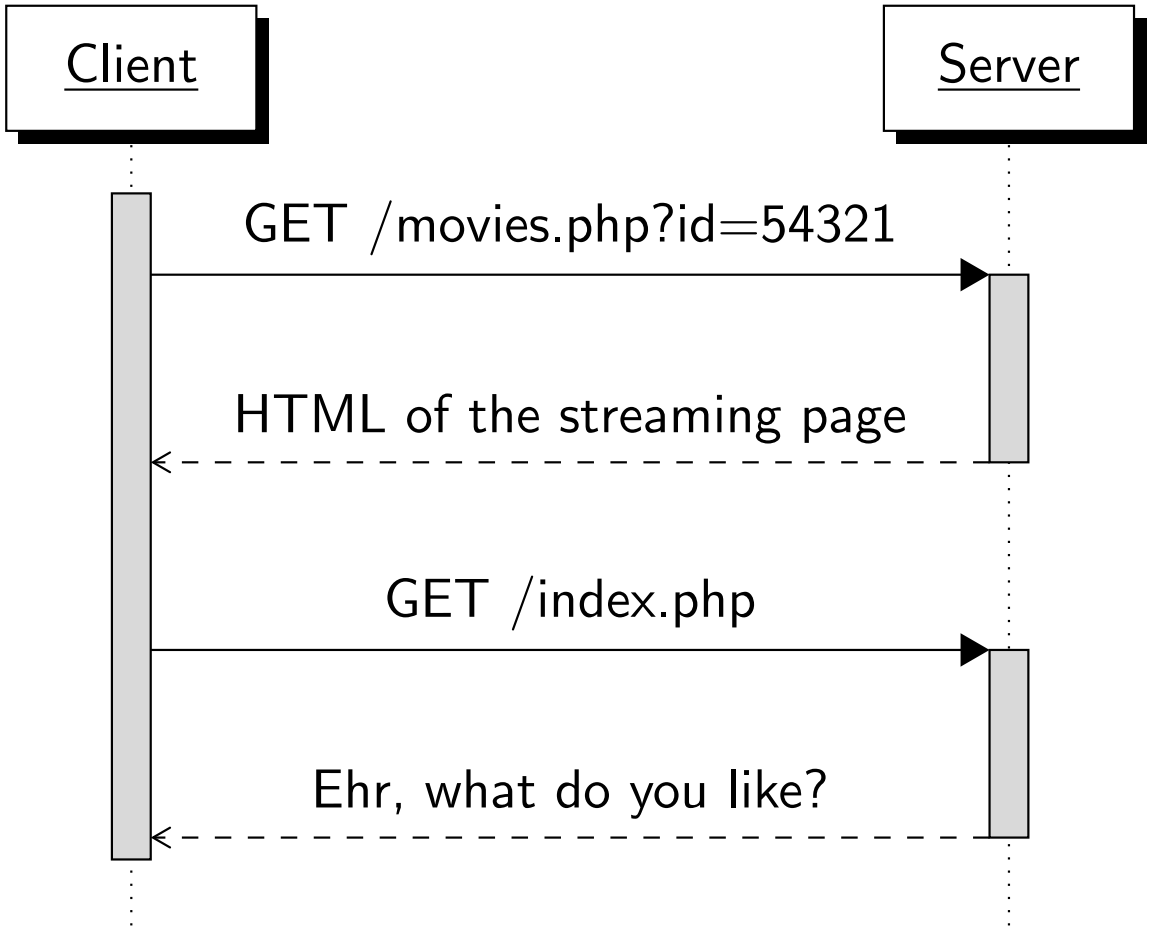
Gli unici status code con un significato speciale per il browser sono i redirect, il cui target è specificato nel `Location` header: il browser processa la risposta ed invia subito una nuova richiesta al target.

Gestione dello Stato in HTTP

HTTP è un protocollo senza stato, cioè due richieste HTTP inviate da uno stesso client risultano sempre "scorrelate" agli occhi del server:

- scelta di design fatta per assicurare la scalabilità
- non è possibile usare la conoscenza relativa alla prima richiesta nella gestione della seconda richiesta
- non è possibile capire che due richieste provengono dallo stesso client, rendendo quindi impossibile la sua autenticazione

Gestione dello Stato in HTTP



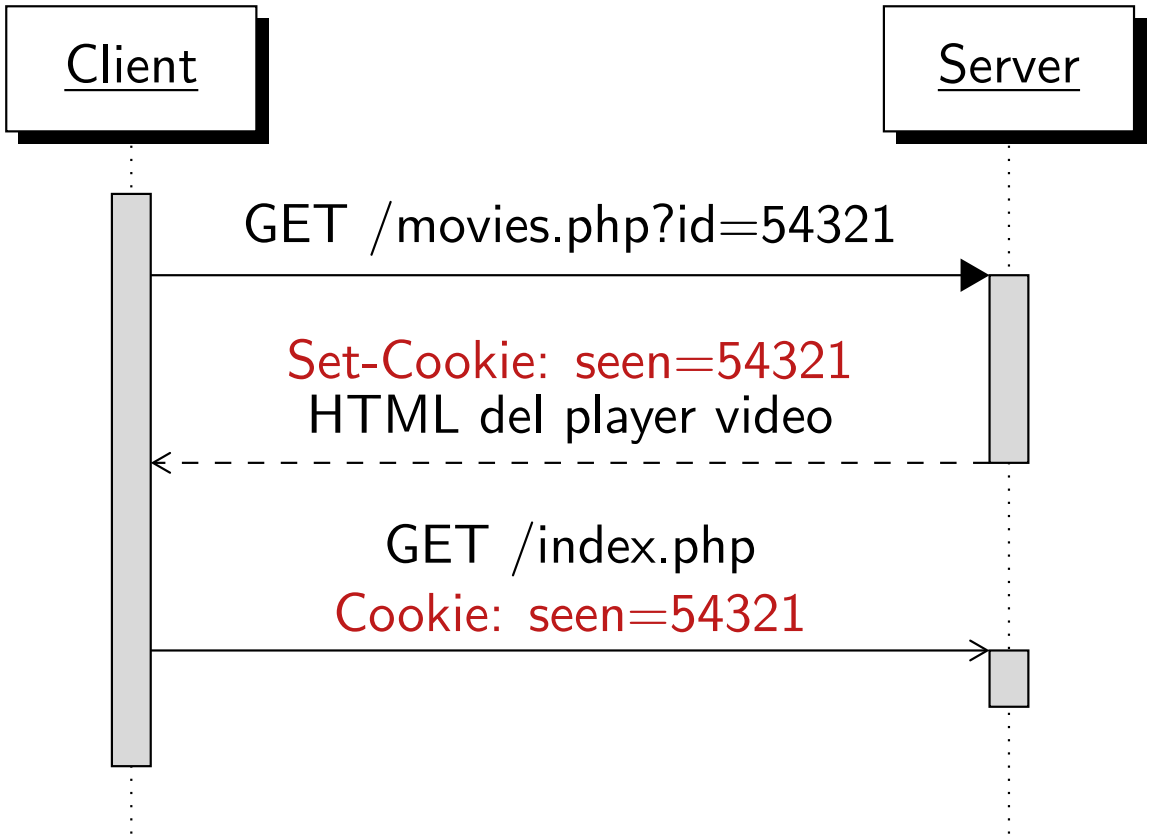
Cookie

La gestione dello stato in HTTP è delegata al client per mezzo di cookie, cioè dati nella forma chiave = valore:

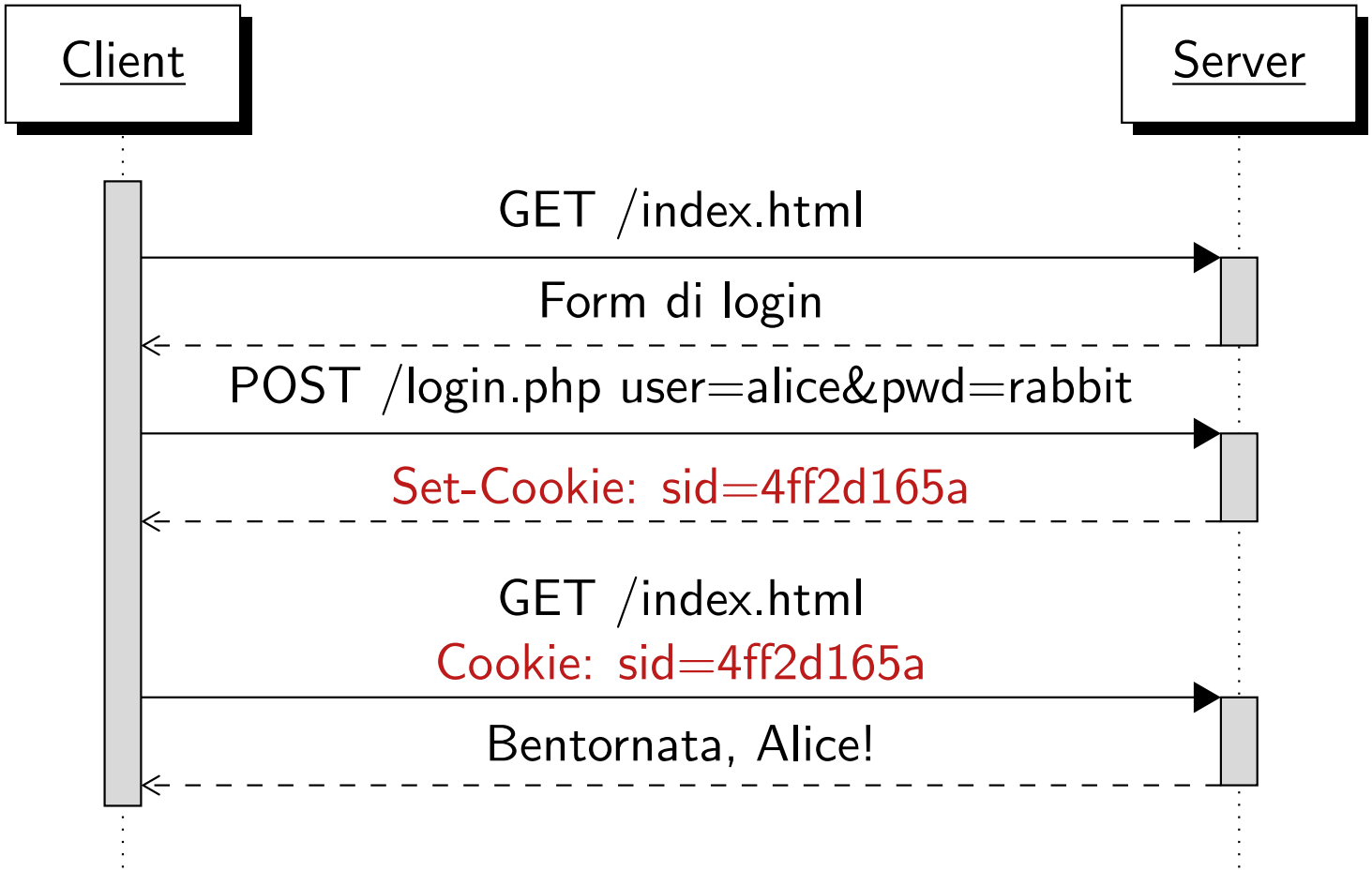
- generati dal server ed impostati nel client attraverso una risposta HTTP tramite l'header `Set-Cookie`
- allegati dal client in ogni richiesta HTTP verso il server che li ha impostati tramite l'header `Cookie`

I cookie sono opachi, cioè il server può salvare ciò che vuole in un cookie, scegliendo una codifica arbitraria. E' compito dello sviluppatore utilizzare tale informazione nel modo corretto.

Cookie: Esempio



Autenticazione tramite Cookie



Sicurezza di HTTP

HTTPS è la controparte sicura di HTTP:

- variante cifrata tramite l'utilizzo di Transport Layer Security (TLS)
- assicura la confidenzialità e l'integrità dei messaggi
- consente l'autenticazione del server tramite certificati firmati

Ormai più diffuso rispetto ad HTTP secondo dati forniti da Mozilla.

Attenzione!

HTTPS è necessario, ma certamente non sufficiente, per la sicurezza di un'applicazione web!

Presentation Tier: HTML + CSS + JavaScript

Tecnologie chiave del presentation tier:

- HTML è un linguaggio di markup utilizzato per la strutturazione di contenuti all'interno di un documento
- CSS è un linguaggio utilizzato per descrivere la presentazione di contenuti strutturati come documento HTML
- JavaScript è un linguaggio di scripting eseguito lato client, che viene utilizzato per lo sviluppo di web application fortemente interattive

HTML

Linguaggio di markup basato su elementi, che definiscono il documento in termini di una struttura ad albero (DOM tree):

- ciascun elemento è descritto da un tag di qualche tipo (intestazione, paragrafo, immagine...), normalmente aperto e chiuso
- il documento è racchiuso fra i tag `<html>` e `</html>`, che vanno a definirne la radice
- ogni elemento può avere degli attributi, che ne definiscono proprietà aggiuntive (es. URL di un'immagine)

Tutorial: `https://www.w3schools.com/html/default.asp`

CSS

Linguaggio basato su mapping fra selettori e dichiarazioni (regole), che definiscono come determinati elementi HTML vanno visualizzati:

- ciascun selettore identifica uno o più elementi HTML
- ciascuna dichiarazione associa proprietà a valori, es. `color:blue`
- quando esistono più regole per uno stesso elemento HTML, viene valutata la loro priorità in base al tipo di selettore ed a dove sono state definite (a parità di priorità, si utilizza l'ultima regola letta):
  1. CSS inline: attributo `style` dell'elemento
  2. CSS interno o esterno: elementi `<style>` o `<link>` in `<head>`
  3. browser default
- alcune proprietà possono essere ereditate, es. `color`

Tutorial: `https://www.w3schools.com/css/default.asp`

W3.CSS

W3.CSS è un framework CSS estremamente semplice da usare ed una valida alternativa al più famoso Bootstrap:

- insieme di classi con regole CSS predefinite e gradevoli alla vista
- sistema a griglia per posizionare gli elementi: griglie predefinite e possibilità di definire nuove griglie (righe da 12 colonne)
- layout responsive: supporto per schermi piccoli, medi e grandi, in cui ogni classe scala verso l'alto
- numerosi temi gratuiti disponibili online

Tutorial: `https://www.w3schools.com/w3css/default.asp`

Checkpoint

Concetti Chiave

- Gli ingredienti fondamentali del protocollo HTTP
- HTML per la strutturazione dei contenuti
- CSS per la presentazione di contenuti strutturati

Materiale Didattico

DBS: Sezione 9.1 + i tutorial indicati nelle slide.