Appunti Progettazione Web

Luca Seggiani
2024

1 Lezione del 23-09-24

1.1 Introduzione

Il corso di progettazione web tratta i seguenti argomenti:

- Il linguaggio HTML 5.0
- I Cascading Style Sheet (CSS)
- Programmazione lato client: JavaScript (ECMAScript)
- Programmazione lato server: PHP
- Il protocollo HTTP

1.2 Storia del Web

Il Web, o World-Wide Web, è un'applicazione di Internet (come lo sono la posta elettronica, il File Transfer Protocol (FTP), ecc...).

1.2.1 Reti a commutazione

Le reti telefoniche (vecchio stile) forniscono un buon modello per il funzionamento di Internet. Le vecchie reti telefoniche erano reti a **commutazione di circuito**, ergo stabilivano una connessione fra due utenti attraverso un circuito fisico. Il filo di rame fisico che collegava i due utenti era riservato a loro, e non poteva servire nessun'altro durante il tempo della chiamata. Questo sistema aveva delle chiare problematiche:

- La connessione va stabilita e mantenuta per la durata della chiamata;
- C'è difficoltà a mantenere più chiamate simultaneamente;
- Si spreca banda in quanto anche i silenzi vengono trasmessi.

Un'alternativa è una rete a **commutazione di pacchetto**. La rete ARPANET non usava una rete a commutazione di circuito, ma una rete a commutazione di pacchetti. In una rete di questo tipo, il messaggio viene diviso in pacchetti, che possono prenderere strade diverse per raggiungere il destinatario. Un pacchetto contiene informazioni sul mittente, il destinatario, e l'ordine del pacchetto stesso nel messaggio trasmesso.

Ad esempio, se Aldo vuole inviare a Bruno il messaggio "Didgeridoo è una parola difficile da dire", questo messaggio potrebbe essere diviso nei pacchetti:

- Aldo | Bruno | 1 | "Didgeridoo"
- Aldo | Bruno | 2 | "è una parola"
- Aldo | Bruno | 3 | "difficile da dire"

e questi pacchetti tramessi attraverso la rete dal mittente al destinatario.

Le reti a commutazioni di pacchetti sono più efficienti in quanto:

- Sono più robuste;
- Non richiedono la totalità delle risorse di trasferimento finché si mantiene la conversazione.

Internet nasce dal progetto ARPANET, e adotta questo tipo di design.

1.2.2 Protocollo TCP/IP

Uno dei primi problemi sorti nella progettazione di Internet è stato quello di unificare le regole di dialogo fra più reti. Una prima soluione a questo problema, sorta nel 1981, è stata il Transmission Control Protocol / Internet Protocol (TCP/IP).

1.2.3 Web

L'invenzione del Web si attribuisce solitamente a Tim Berners-Lee. Dalla sua concezione iniziale ad oggi, le caratteristiche fondamentali del sistema sono rimaste invariate:

- Un Unique Resource Locator (URL) per identificare e locare risorse sul Web;
- Il protocolo HTTP per descrivere come le richieste e le risposte operano;
- Un programma, chiamato **server**, che risponde alle richieste HTTP;
- Il linguaggio HTML per pubblicare documenti;
- Un programma, chiamato **browser** (che fa la parte del **client**), che crea richieste HTTP dagli URL, e visualizza l'HTML che riceve.

Poco dopo la sua fondazione, Berners-Lee aiutò a fondare il Word Wide Web Consortium (W3C), che ha facilitato la crescita della rete attraverso la creazione di standard. Questo è stato permesso dalla decisione del CERN di rendere tutto il codice sorgente e i protocolli web sviluppati in questo periodo, effettivamente liberi.

Alcuni dei vantaggi delle applicazioni Web sono:

- Accessibilità da parte di ogni computer connesso a Internet;
- Utilizzabile con più browser e sistemi operativi;
- Più semplici da aggiornare;
- Archiviazione dei dati centralizzata.

alcuni contro, invece, sono:

- Richiesta di avere una connessione internet attiva.
- Problemi di sicurezza, dati dalla trasmissione di dati sensibili attraverso Internet;
- Problemi di inaderenza agli standard di alcuni siti su alcuni browser;
- Accesso limitato al sistema operativo.

1.2.4 Intranet

Un'intranet è una rete internet limitata ad un'organizzazione o un'azienda. A differenza delle reti internet, le reti intranet sono private e limitate agli utenti che le usano (impiegati o clienti). Il controllo sulle connessioni in entrata e in uscita di una rete internet o intranet è solitamente eseguito da un **firewall**.

1.2.5 Siti Web statici

Un sito web statico è formato da pagine HTML che appaiono identiche a tutti gli utenti in tutti i momenti.

Storicamente, questi contenuti venivano mantenuti dal cosiddetto webmaster, che doveva solo modificare il codice in HTML e, eventualmente, immagini o altri contenuti multimediali.

Il principio di funzionamento di un sito statico è il seguente:

- L'utente (il client) richiede una pagina al server;
- Il server risponde alla richiesta erogando la pagina desiderata;
- L'utente riceve la pagina e la visualizza.

1.2.6 Siti Web dinamici

Dopo alcuni anni dall'invenzione del Web, alcuni siti iniziarono a diventare più complicati e ad usare programmi che giravano sui web server per creare contenuti in maniera dinamica.

Secondo questo modello, il principio di funzionamento di un sito statico è il seguente:

- L'utente (il client) richiede una pagina al server;
- Il server risponde alla richiesta erogando la pagina desiderata e eventualmente riempendola di informazioni generate da script, programmi, ecc... eseguiti sul momento;
- L'utente riceve la pagina e la visualizza.

1.2.7 Web 2.0

Web 2.0 è una buzzword che si riferisce principalmente a due cose:

- Per gli utenti, Web 2.0 significa applicazioni interattive dove si possono consumare ma anche creare nuovi contenuti;
- Per gli sviluppatori, Web 2.0 significa un nuovo modo di creare siti Web dinamici: la logica di programmazione si spostava dal server al client, attraverso il linguaggio Javascript. Questo introduceva diversi problemi per quanto riguarda l'esecuzione di comunicazioni asincrone.

1.3 Protocolli Internet

Un protocollo è una serie di regole che vengono usate da più calcolatori che comunicano fra loro. I protocolli TCP/IP vennero inizialmente pensati come uno stack di quattro livelli, anche se alcune astrazioni dividono questo stack ulteriormente in quattro o cinque livelli. Vediamo quindi i quattro livelli fondamentali, dal basso verso l'alto:

1. **Link Layer:** si occupa della trasmssione fisica di bit di informazione.

   Esempio: **MAC**;

2. **Internet Layer:** stabilisce connessioni, routing e indirizzamento.

   Esempio: **IPv4**, **IPv6**;

3. **Transport Layer:** si assicura che le trasmissioni arrivano in ordine e senza errori.

   Esempio: **TCP**, **UDP**;

4. **Application Layer**: protocolli che permettono alle applicazioni di interagire con il Transport Layer.

   Esempio: **HTTP**, **FTP**, **POP**.

1.3.1 Link Layer

Il link layer permette il controllo sull'hardware che stabilisce un collegamento con i mezzi di trasmissione cablata (vedi Ethernet) o wireless (vedi Wi-Fi), ovvero il Media Access Control (MAC), e fornisce gestione di errori e di flusso, ovvero il Logical Link Control (LLC).

1.3.2 Internet Layer, protocollo IP

Il protocollo IP identifica destinazioni attraverso indirizzi IP. Ogni dispositivo connesso a internet ha un indirizzo IP proprio, ovvero un codice di 32 bit che lo identifica univocamente.

In particolare, gli indirizzi IPv4 sono quelli del protocollo TCP/IP originale. La rappresentazione comune degli indirizzi IPv4 (192.168.0.1) racchiude fra punti interi di 8 bit per volta dei 32 dell'indirizzo.

Gli indirizzi IPv6 appartengono ad un protocollo aggiornato, attualmente in corso di adozione, che adottano 8 interi da 16 bit. Solitamente gli interi del protocollo IPv6 vengono scritti in esadecimale.

L'indirizzo IP viene solitamente assegnato da un Internet Service Provider (ISP). Su reti locali, più computer possono condividere un'indirizzo IP, attraverso il NAT. NAT sta per Network Address Translation. All'interno della rete locale, ogni dispositivo usa un indirizzo diverso e privato (10.0.0/24). In uscita dalla rete locale, tutti i datagrammi hanno lo stesso indirizzo NAT IP di mittente, e diversi numeri di port. Questo è permesso da un meccanismo interno al ruoter NAT che converte automaticamente indirizzi IP e numeri di porta in indirizzi IP locali, e viceversa.

1.3.3 Transport Layer, protollo TCP

L'IP ha diverse limitazioni:

- La consegna dei pacchetti non è garantita;

- I pacchetti potrebbero arrivare in disordine;
- La comunicazione avviene fra dispositivi;
- Nessun controllo sul flusso o sulle congestioni.

Il Transport Layer si assicura che ogni trasmissione arrivi in ordine e senza errori, attraverso dei meccanismi di sicurezza:

1. Prima i dati sono divisi in pacchetti formattati secondo il protocollo TCP;
2. In seguito, ogni pacchetto viene restituito al mittente attraverso l'ACK (acknowledge).

In questo modo, il mittente sa se i suoi dati sono stati effettivamente ricevuti, e può reinviare i pacchetti perduti.

Il protocollo TCP fornisce una serie di altre funzionalità: la gestione del flusso e della congestione. Questi meccanismi riguardano la diminuzione della frequenza di trasmissione in caso di sovraccarico del destinatario.

1.3.4 Application Layer

L'application layer è un livello di astrazione. Fanno parte di questo livello il protocollo di trasferimento di pagine web, ovvero l'Hypertext Transfer Protocol (HTTP), i protocolli di trasmissione di file, ovvero il File Transfer Protocol (FTP), o i protocolli di trasmissione di posta elettronica, come il Post Office Protocol (POP), fra gli altri.

1.4 Modello client-server

Nel modello client-server, i dispositivi si dividono in client e server. Il server è attivo e pronto ad erogare servizi, mentre il client si occupa di fare richieste a cui il server dovrà adempire. Questo meccanismo forma il cosiddetto request-response loop.

1.4.1 L'alternativa peer-to-peer

In una rete peer-to-peer non esiste un unità centrale (il server), ma ogni nodo è in grado di inviare e ricevere indipendentemente dagli altri (peer).

Questo sistema è molto comodo per la trasmissione in massa di dati senza la possibilità di arrestare il processo a partire da una sola macchina (vedi Napster).

1.4.2 Tipi di server

Prima, abbiamo mostrato il server come una macchina singola. In verità, il server può essere formato da più macchine che si dividono il carico delle richieste dei client, e che rispondono dallo stesso URL.

Un'altra alternativa sono le cosiddette **server farm**. Una server farm distribuisce il carico delle richieste (attraverso i **load balancer**). Le server farm forniscono inoltre **failover redundancy**, ovvero la sicurezza che il servizio non si arresti anche nel caso di fallimento di singole macchine.

Le server farm sono tipicamente collocate all'interno di specifiche strutture dette **data center**. Non è inusuale archiviare lo stesso sito su più macchine, in parallelo. Può accadere anche il contrario: più pagine possono essere archiviate sullo stesso server.

1.5 Nozioni sull'infrastruttura

Oltre ai server, che abbiamo visto, altri componenti vanno a formare la rete Internet: innanzitutto il modem (che può essere di tipo Digital Subscriber Line (DSL) se sfrutta la linea telefonica), che funge da ponte fra la rete interna a aziende o abitazioni, e la rete esterna, gestita da un Internet Service Provider (ISP) o un'altra autorità.

Da qui il router, che si occupa di indirizzare i pacchetti secondo determinate tabelle chiamate routing tables, che includono per tutti gli IP di destinazione noti il prossimo passo (hop) da effettuare, ovvero il prossimo IP a cui inviare il pacchetto.

I cavi ottici arrivano infine all'head-end dell'ISP, dove i dati vengono gestiti da apparecchiature come il Cable Modem Termination System (CMTS), o il Digital Subscriber Line Access Multiplexer (DSLAM) per le tramissioni DSL.

Da qui i dati potrebbero dover viaggiare oltre, tipicamente arrivando ad altre reti, che possono avere estensione anche locale. Per rendere più veloci le tramissioni fra più reti, oggi si adoperano gli Internet Exchange Point (IX o IXP), che permettono a più reti di unirsi in strutture condivise.

Spesso, sopratutto di recente, le grandi compagnie decidono di installare la loro infrastruttura (server e data center) il più vicino possibile agli IXP, per velocizzare la trasmissione portando i dati più vicini agli utenti.

2 Lezione del 25-09-24

2.1 Domain Name System

Il Domain Name System (DNS) è il sistema che usiamo per tradurre gli indirizzi IP in nomi simbolici, più familiari ad utenti umani. L'idea è quella di ricavare un indirizzo IP a partire dal DNS, stabilire una comunicazione col protocollo IP, e trasmettere una pagina web HTML.

La risoluzione di un DNS viene effettuata da un Domain Name Server, solitamente gestito dall'ISP.

I nomi DNS sono formati da più livelli (**domini**) separati da punti, ad esempio `server1.www.pippo.com`. L'ultimo dominio si chiama Top Level Domain (TLD), e da lì in poi, da destra verso sinistra, si assegnano numeri progressivi da 2 (Second Level Domain, Third Level Domain, ecc...). Il TLD è più generico, l'ultimo dominio il più specifico.

Esistono più tipo di TLD:

- Generic top-level domain (gTLD)
  - Unrestricted (.com, .net,. org, ...)
  - Sponsored (.gov, .mil., edu, ...)
- Country code top-level domain (.it, .us, ...)

2.1.1 Registrazione di nomi

I nomi di dominio vengono assegnati e gestiti da particolari organi detti **registri**. Per registrare un dominio ci si rivolge ai registri o agenzie intermedie, e si forniscono alcune informazioni particolari di natura amministrativa.

2.1.2 Risoluzione di nomi DNS

La traduzione dal nome simbolico a quello numerico (cioè l'IP) viene effettuato dai DNS resolver. Solitamente, in verità, i DNS noti sono memorizzati nella cache del nostro browser. Nel caso un DNS non sia trovato nella cache, si cerca in una componente apposita del sistema operativo. Se nemmeno qui si trova il DNS desiderato, si fa una richiesta al server DNS dell'ISP.

A questo punto pure il server DNS controlla nella sua cache. Nel caso nemmeno il server DNS trovi il DNS desiderato, esso si rivolge a un Root name server, ovvero uno dei 13 server delegati a quest'operazione, che restituirà l'indirizzo IP per il dominio di livello più alto del DNS. Questa operazione si ripete su ogni livello del dominio per risolvere il DNS fino al livello più profondo.

2.2 Uniform Resource Locator

L'Uniform Resource Locator (URL) è un sistema per dare nomi a ogni file contenuto all'interno di uno web server. L'URL ha forma:
`http://www.pippo.com/index.php?page=17#article`
Prima si specifica il protocollo (`http://`), poi il dominio (`www.pippo.com`), il percorso o *path* (`index.php`), la stringa di query (?page=17) e il frammento (`#article`).

2.2.1 Protocollo e dominio

La prima parte dell'URL indica il protocollo usato, e la seconda il dominio, che può essere un DNS o un'indirizzo IP, e su cui si puo specificare dopo "::" il numero di porta. Entrambe le parti sono case insensitive.

I numeri di porta di default sono ad esempio 21 per il protocollo ftp, e 80 per il protocollo http.

2.2.2 Stringa di Query

Una stringa di query serve a passare informazioni dall'utente al server. Sono codificati come coppie chiave-valore delimitate dal carattere **&** e precedute dal carattere **?**.

2.2.3 Uniform Resource Identifier

Un Uniform Resource Locator, come un Uniform Resource Name (URN), fa parte di una categoria più ampia detta Uniform Resource Identifier (URI). In particolare, si può dire che:

- Un **URI** identifica una risorsa senza necessariamente contenere particolari informazioni su come trovarla;
- Un **URL** identifica una risorsa e specifica come trovarla;
- Un **URN** fa il lavoro di un URI ma con regole molto più stringenti.

2.3 Richieste HTTP

L'HTTP è il protocollo usato per ottenere pagine web da web server. Quando si accede ad un sito con il browser, questo invia al web server una richiesta HTTP, dove specifica il DNS cercato (più server possono gestire più siti web), richiede una certa risorsa, e

trasmette informazioni su di sé (tipo di browser, encoding e lingue accettate, ecc...), ad esempio:

```
GET /index.html HTTP/1.1
Host: pippo.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/20100101
    Firefox/15.0.1
Accept: text/html,application/xhtml+xml
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Connection: keep-alive
Cache-Control: max-age=0
```

dove si nota la richiesta `keep-alive` di mantenere aperta la connessione, e la richiesta `max-age=0` sulla cache che chiede al server di fornire risorse aggiornate.

A questo punto il server risponde alla domanda fornendo informazioni sul tipo di server, sul formato della risorsa inviata, e la risorsa stessa:

```
HTTP/1.1 200 OK
Date: Mon, 25 Sep 2024 02:08:49 GMT
Server: Apache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 4538
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
...
```

segue la pagina web vera e propria.

2.3.1 Parsing e richieste successive

Solitamente un file HTML contiene ulteriori riferimenti ad altre risorse (stylesheet, altri file html, immagini, ecc..). Per ogni nuova risorsa che si rende necessaria, si fa una nuova richiesta al web server.

2.3.2 Metodi di richiesta

I tipi di richiesta vengono anche detti **metodi**. Esistono più metodi, fra cui:

- **GET:** richiede una risorsa dal server;
- **POST:** invia informazioni al server, ad esempio per trasferire un form.
- **HEAD:** richiede solo l'intestazione o *header* della risorsa, ad esempio per controllare se ha già la versione più recente in cache;
- **PUT:** aggiorna o rimpiazza una risorsa ad un dato URL. Se non esiste, la crea;
- **DELETE:** Rimuove una risorsa a un dato URL.
- **CONNECT:** Stabilisce una connessione col server. Spesso è utilizzato per connessioni SSL (HTTPS);
- **TRACE:** Risponde con la stessa richiesta. Usata per motivi di debug;
- **OPTIONS:** Descrive le opzioni di comunicazione per la risorsa interessata. Utile per trovare quali metodi HTTP sono supportati dal server.

2.3.3 Codici di risposta

HTTP prevede dei codici di risposta alle richieste:

- **1##:** risposte informative, che assicurano il proseguimento dell'operazione;
- **2##:** codici di successo operazione, ad esempio:
  - 200 "OK"
- **3##:** codici di ridirezione (risorse spostate), ad esempio:
  - 301 "Risorsa spostata permanentemente"
  - 304 "Ridirezione temporanea"
- **4##:** errori lato client, ad esempio:
  - 400 "Richiesta malformata"
  - 401 "Non autorizzato"
  - 404 "Non trovato"
  - 414 "URI richiesto troppo lungo"
- **5##:** errori lato server, ad esempio:
  - 500 "Errore server interno"

2.4 Web server

Un web server è un computer che risponde a richieste HTTP. Sul web server gira il cosiddetto **stack**, che comprende il software del server:

- Il sistema operativo;
- Il software web server;
- Un database;
- Un linguaggio di scripting;
- ...

Solitamente ci si riferisce agli stack comuni:

- **LAMP:** Linux, Apache web server, MySQL database, PHP;
- **WISA:** Windows, IIS web server, SQL Server database, ASP.NET.
- **XAMP:** un pacchetto fornito da Apache. XAMPP Apache, MariaDB, PHP, Perl-
- **WAMP:** Windows, Apache web server, MySQL databse, PHP.

2.5 HTML

L'HTML non è un linguaggio di programmazione, ma un linguaggio di **markup**, ovvero usato per dare una struttura a dei documenti.

L'HTML è gestito dal W3C, che produce raccomandazioni (chiamate anche **specifiche**). Nel 1998 il W3C ha proposto uno standard diverso, detto XHTML, che cercava di risolvere alcuni dei problemi dell'HTML, adottando regole di sintassi più severe e basate sull'XML.

Le regole principali dell'XHTML sono:

- I nomi dei tag sono in lower case;
- Gli attributi sono sempre fra virgolette;
- Tutti gli elementi devono avere un elemento di chiusura (o chiudersi da soli).

Per aiutare gli sviluppatori, due versioni di XHTML furono create: XHTML 1.0 Strict e XHTML 1.0 Transitional.

- La versione **strict** doveva essere renderizzato da un browser usando le regole di sintassi più severe;
- La versione **transitional** aveva delle regole più rilassate, ed era pensato come strumento per la transizione temporanea da HTML a XHTML.

2.5.1 Validatori

Parte degli sforzi di questi anni hanno dato luogo allo sviluppo di **validatori**, ovvero strumenti atti a validare che un dato documento HTML rispetti determinati standard.

Nella metà degli anni 2000, XHTML 2.0 propose un cambiamento sostanziale all'HTML, che abbandonava la compatibilità con HTML e XHTML 1.0.

In risposta, sì formo un comitato detto Web Hypertext Application Technology Working Group (WHATWG) all'interno del W3C. Il lavoro di questo comitato ha portato all'ultima versione, l'HTML5, caratterizzato da:

- Specifica non ambigua di come i browser dovrebbero gestire il markup invalido;
- Un framework aperto e non prioritario (JavaScript) per lo scripting;
- Compatibilità con il web già esistente.

2.6 Sintassi dell'HTML

I documenti HTML sono composti da contenuti testuali ed elementi HTML.

2.6.1 Elementi

Gli elementi HTML sono formati da:

- Il nome dell'elemento o **tag** racchiuso fra freccette (< e >);
- Eventuali **attributi**;
- Il contenuto dentro il tag.

Ad esempio:

```
<a href="http://www.pippo.com"> Pippo </a>
```

2.6.2 Elementi vuoti

Un elemento può essere vuoto, ovvero non contenere contenuti. In questo caso si può adottare un *trailing slash* opzionale:

```
<img src="pluto.gif" alt="pippo" />
```

2.6.3 Annidamento di elementi

Gli elementi HTML sono effettivamente annidati, ovvero possono contenere altri elementi HTML. In questo caso si stabiliscono le solite relazioni padre-figlio.

2.6.4 Markup semantico

L'HTML ha il compito di definire la struttura semantica del documento, e non come questo viene mostrato, ad esempio su più dispositivi. A occuparsi di questo sono i Cascading Style Sheets (CSS).

Questa separazione è utile a più scopi:

- **Mantenibilità:** il markup semantico rende più semplice la modifica di pagine graficamente complesse;
- **Prestazioni:** le pagine semantiche sono più veloci da scrivere e da scaricare, il CSS può essere messo in cache;
- **Accessibilità:** strumenti come le lettura dello schermo sono più semplici da implementare su documenti semantici;
- **Ottimizzazione di motori di ricerca:** il markup semantico rende il sito più semplice da vedere per i motori di ricerca.

2.6.5 Struttura di un documento HTML

Un documento HTML semplice si presenta come:

```
<!DOCTYPE html>
<html>
<head lang="en">
<meta charset="utf-8">
<title>Share Your Travels -- New York - Central Park</title>
<link rel="stylesheet" href="css/main.css">
<script src="js/html5shiv.js"></script>
</head>
<body>
<h1>Main heading goes here</h1>
...
</body>
</html>
```

Vediamo le sue componenti:

- `DOCTYPE` specifica il tipo di documento, in questo caso HTML;
- `html` è un nodo radice, opzionale, da cui partono:
  - `head`, che è la testata della pagina (banalmente il titolo). Si noti come qui è specificato il character set (qui utf-8);
  - `body`, che contiene i contenuti veri e propri del sito.