Programmazione web e servizi digitali

# Lezione 1: Introduzione

Programma:

* Html, css
* Xml, xhtml
* Php (lato server)
* FrameWork:
  + **Bootstrap** (html e css per fare pagine responsive = differenza tra dispositivi gestita)
  + **Laravel** e Eloquient (ORM) (x PHP)
    - ORM= Object-Relational Mapping: librerie x gestire le relazioni con i DB Relazionali.
* Lato Client:
  + Javascript (jQuery)
  + RIA= rich internet applications (AJAX, Fetch, mette in collegamento Client e server, aumenta la performance)
* servizi
  + Architetture orientate ai servizi
  + Basate su XML
  + RESTful servicies (usa json)

WEB:

* 1.0: web classico, siti dell’uni e delle istituzioni, siti mastodontici, offrono contenuti ai consumer;
* 2.0: social web, portato avanti dagli utenti (Reddit, Facebook);
* 3.0: personalizzabile, super custom per l’utente, dipende particolarmente dall’autentificazione, “web semantico”.

ESAME: piccolo esame scritto programmazione su Laravel (8) + Elaborato (23)

## HTTP e HTML

Architettura client-server

Differenza: le app sono runnate parte server e parte client

Bisogna sapere cosa và dove.

Ogni richiesta client al server richiede una risorsa. Ma esiste la programmazione lato client (javascript) (es: si possono fare controlli) e non è necessario contattare sempre il server.

Sul Server ci sono le risorse.

**HTTP**: protocollo di richiesta e risposta usato dai server web.

* Header: si utilizza per informazioni (status, browser ... )
* Payload: contiene un file HTML
  + Se statica basta metterla sul client
  + Se dinamica può essere che bisogna mettere in comunicazione javascript e PHP

HTTP: è stateless! Quindi qualsiasi gioco di richiedi e ottengo, è indipendente da altri. Possibili problemi con la parte di autentificazione, serve gestire uno stato (nel server! Dato che PHP è stateless).

**Richiesta**: URL, uniform resource locator

* http:// --- protocollo
* Server.unibs.it:80 --- indirizzo:porta
* /albero delle directory/file.html

Noi lavoreremo con localhost (127.0.0.1) come address.

**RISORSA: (**ogni risorsa ha un URL)

* File HTML
* Foglio di stile (CSS)
* Contenuti multimediali
* PHP: codice, script, eseguito lato server per richiesta client; l’output è una pagine HTML, modificata dinamicamente dal PHP.
* Codice Javascript: script lato client, script presi dal server e caricati sul client per eseguirli sul browser client.

**HTML** (Hypertext markup language):

* Markup language: per esporre contnuti organizzandoli, annidandoli e raggruppandoli
* Organizza attraverso dei TAG: <tag></tag>;
  + Sempre apertura e chiusura
  + Se non c’è contenuto: <tag/>
  + Annidamento: divere profondità ma bisogna rispettare l’annidamento
    - Per poter trasformate in un ALBERO!
    - Albero: per fare il **DOM**: document object model, per controllare con javascript l’HTML lato clietn
* I TAG sono predefiniti
* Danno al browser info per visualizzare
* ES:
  + <html>
  + <head><title> “sito” </title></head>
  + <body> ... < /body>
  + </html>
* Nei tag ci sono gli **attributi,** nel tag di apertura. (**attributo1=”valore”**)
  + Tag image (che non ha contenuto): ha attributo source che specifica l’immagine da caricare
* **Link** (anchor): <a href=” risorsa (UML)”> </a>;
  + per mettere hypertexts (links, *risorse*)
  + Possibile specificare dove apre il link con attributo *target*
* Tipi di tag:
  + In Line: non va a capo: ciao <bold> pirla</bold> --> ciao **pirla**
  + Blocco: fa un nuovo paragrafo (va a capo)
  + Per applicare stili: <span> e <div> tag
  + Per controllare il flusso del testo
  + Per controllare la formattazione del testo
  + Per controllare la formattazione di base
  + Per controllare il flusso del testo: per mettere etichette (come un # che poi potrei cercare)
  + Liste: elenchi puntati:
  + <ul>: unordered list
  + <li> list item
  + Tabelle
  + Links
  + Interazione con utente (input, button)
* Attributi Tag:
  + id: ID unico per riferirsi agli elementi negli script
  + **class**: per identificare classi di tag: gruppi di id.
    - Id e **class** sono la base su ci lavorano i framework, che mettono lo stile loro (in base alle class)
  + style: serie di regole CSS.
  + title
  + lang: per dire in che lingua è la pagina
  + Molti altri, sporattutto per programmazione ad eventi (onclick, onmouseup ...)
    - Serve un EeventListener e un Handler
    - L'utente scatena tantissimi eventi.
* Tabella: sono l’inferno! Solo dio sa perché esistono e son fatte così
  + <tr> table ROW
  + <th> o <td> per le celle dentro le ROW
* Immagini: <img>
  + Attributo src=URL
    - Può essere un riferimento assoluto o relativo
      * Relativo. Si lavora con le cartelle, meglio perché è possibile spostarlo facilmente
  + Attributo alt=URL di riserva
* Form: per far interagire con il web e il server
  + Attributi:
    - Name: per dare nomi agli input dell’utente, così sul server vedi coppie (Nomi, valori)
    - Action: la risorsa da chiedere
    - Method: metodi HTTP da usare x chiedere la risorsa

# Lezione 2: CSS

Per dare uno stile ad HTML. CSS sono regole di formattazione

Raccomandazione: HTML è il più asettico possibile e si modifica lo stile con CSS. Perché? Come sempre: modulazione e per poter applicare velocemente ad ogni pagina e non dover modificare in tutte le parti manualmente.

CSS: file di testo fatto da tante linee:

* Selettore {proprietà: valore}
  + Selettore: modo di individuare la parte dell’HTML (modo più facile: indicare TAG
    - es: h1,h2 {color:green} );
    - Possibile indicare (usando la struttura del DOM:
      * Es: tutti i qualcosa in qualcosa
        + Descendant selector: h1 p {prop:valore}
        + Child selector: h1 > p {}
        + Class selector: .value (seleziona gli elementi con calss=value)

h1.value

* + Proprietà: tutte le caratteristiche di stile
    - Es: {text-align:center; color:blue}
    - Dividi con un ;
  + Valore: il valore dell’attributo modificato
    - Si usa “ “ se sono +2 parole con spazio
* Tutto questo funziona perché c’è annidamento (il DOM, l’albero dell’html)
  + Quindi ci sono i genitori e figli; antenati e discendenti; fratelli.
  + Esiste l’ereditarietà! Ma può essere sovrascritta dai discendenti

**Collegare HTML e CSS:**

<head>

<link rel=”stylesheet” type=”text/css”   
 href=”mystyle.css” />

</head>

* rel= obbligatorio: descrive relazione tra documento e file collegato
* href= obbligatorio: URL del CSS
* type: opzionale ma meglio metterlo per sicurezza: identifica il MIME
* media: opzionale (poco usato dal prof): identifica il supporto (sreen, print …)

Puoi modificare anche le tabelle! Ricorda di farlo in percentuale però.

VANTAGGI di CSS:

* più facile gestire la grafica,
* divide HTML che fornice dati e CSS che fa grafica
* Stile uniforme tra pagine diverse
* Effetto cascata delle modifiche
* Accessibilità: più facile interagire col sito.