



Università degli Studi di Bologna
Scuola di Ingegneria

Tecnologie Web T
A.A. 2020 – 2021

Esercitazione 1
URI, HTTP, HTML, CSS

1

Agenda

- URL e Protocollo HTTP
 - prove ed esempi

- HTML e CSS
 - corrispondenza tra elementi visualizzati e codice sorgente
 - “ispezione” del contenuto di una pagina
 - “ispezione” degli stili applicati agli elementi di una pagina
 - esempi da ricreare

2

URL... non solo pagine Web

- Accesso a una risorsa via HTTP
 - <http://lia.disi.unibo.it/Courses/twt2021-info/>
- Scaricamento della stessa risorsa via FTP
 - <ftp://lia.disi.unibo.it/Courses/twt2021-info/> ????
 - ...o anche...
 - <ftp://ftp.iinet.net.au/debian/debian-cd/>
- Streaming di file multimediali
(possibili client: VLC, Windows Media Player, ...)
 - <mms://151.1.245.36/rt1102.5lq>
- Eccetera, eccetera...
 - http://en.wikipedia.org/wiki/URI_scheme



URL e pagine HTML

- Poiché non disponiamo ancora di un Web Server su cui esercitarci...
 - è necessario aprire le pagine HTML dell'esercitazione di oggi leggendole da file system
- È possibile farlo “manualmente”...
 - tramite i menu a tendina del browser (File → Open → ...)
- Oppure nel modo (appena un po') più “geek”
 - digitando a mano l'URL adeguato nella barra degli indirizzi del browser

Ispezione di codice HTML esistente

- Dopo aver estratto i file presenti nell'archivio "01_TecWeb.zip" dell'esercitazione, nelle directory *html1* e *html2* trovate
 - alcune semplici pagine HTML di esempio
- Per visualizzarle nel browser
 - **quale URL deve essere immesso nella barra degli indirizzi?**
 - attenzione a fare "escaping" dei caratteri speciali (*blank = %20*)
- Per visualizzare il codice sorgente
 - **Basta un qualsiasi editor di testo**
 - Visual Studio Code (raccomandato)
 - Notepad++
 - Gedit
 - Kedit
 - Sublime Text
 - ...

Esercitazione 1

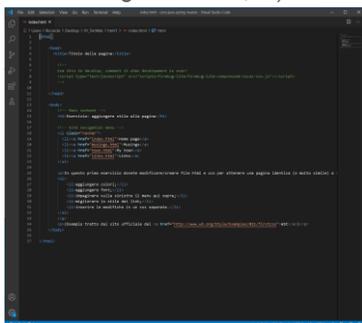
5

5

Editor HTML vs. User-agent

Confrontiamo il "sorgente" HTML della pagina e la sua versione "renderizzata"

- via editor testuale (es: *VS Code*, *notepad++*, *gedit*, *kedit*, ...)
 - IDE (*Eclipse*, *IntelliJ IDEA*, *NetBeans*, ...)
 - direttamente dal browser (→ visualizza sorgente...)
- vs.
- user-agent (es: *Microsoft Internet Explorer*, *Mozilla Firefox*, *Opera*, *Safari*, *Google Chrome*, ...)



Esercizio: aggiungere stile alla pagina

- [Home page](#)
- [Musings](#)
- [My town](#)
- [Links](#)

In questo primo esercizio dovete modificare/creare file html e css per ottenere una pagina identica (o molto simile) a [obiettivo.png](#). Si consiglia di seguire i seguenti passi:

1. aggiungere colori;
2. aggiungere font;
3. impaginare sulla sinistra il menu qui sopra;
4. migliorare lo stile dei link;
5. inserire le modifiche in un css separato.

(Esempio tratto dal sito ufficiale del [W3C](#))

Esercitazione 1

6

6

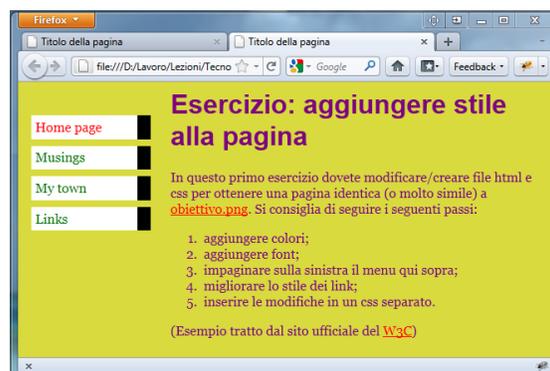
index.html

- Utilizzare **Google Chrome Developer Tools** o **Firefox for Developer** per analizzare/modificare i file
 - uso della funzione **Inspect Element**
 - aggiunta di stili “on-the-fly”
- **Elementi di interesse**
 - **foglio di stile** esterno: `styles/default.css`
 - **ancore interne** (che il browser concatena all'URL corrente) per puntare a specifici elementi della loro stessa pagina
 - **iframe** per includere codice HTML “esterno” (oggi tipicamente usati per permettere applicazioni cross-site → es: Google Maps)

Can you do that? (html1)

Ottenere il seguente risultato:

- aggiungere colori
- aggiungere font
- impaginare sulla sinistra il menu qui sopra
- migliorare lo stile dei link
- inserire le modifiche in un file CSS separato



Can you do that? (html2)

Ottenere il seguente risultato:

- aggiungere gli opportuni stili in un secondo file CSS
- modificare il sorgente HTML solo per includere tale file, inserire gli elementi della form e ottenere l'unione di alcune celle della tabella

Un semplice esempio di pagina HTML

Confrontate il sorgente con il rendering eseguito dal browser!

The screenshot shows a browser window with a blue background. At the top, there are three blue boxes with white text: "una lista", "paragrafi e immagini", and "un iframe contenente una tabella". Below these, the page content is rendered. The first part is a list of elements. The second part is a paragraph of text with a placeholder for a missing image. The third part is another paragraph. The fourth part is an iframe containing a table with 3 rows and 4 columns. The table has a header row with columns A, B, C, and D. The first row contains "cella A1", "cella B1", "cella C1", and "cella D1". The second row contains "cella A2", "contenuto presente all'interno della cella B2", and "contenuto presente all'interno della cella D2". The third row contains "cella A3", "cella B3", "cella C3", and "D3". Below the table, there is a small text box that says "Una form HTML richiede un pulsante di invio!".

Esercitazione 1

9

9

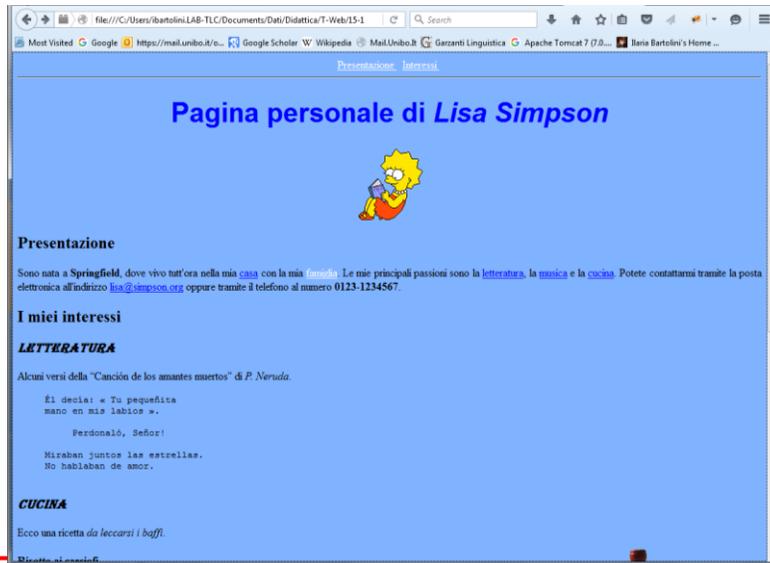
APPENDICE

(altri esempi html/css da cui apprendere...)

10

Esempio "Lisa Simpson": formattazione tipografica

"01_TecWeb.zip", directory *esempi/Simpson*



Esercitazione 1

11

11

Esempio "PosizionamentoSenzaFrame": layout liquidi

"01_TecWeb.zip", directory *esempi/PosizionamentoSenzaFrame*



Esercitazione 1

12

12

Esempio HTML 5 CSS 3: “Le griglie”

“01_TecWeb.zip”, directory `esempi/HTML5CSS3/grid.html`



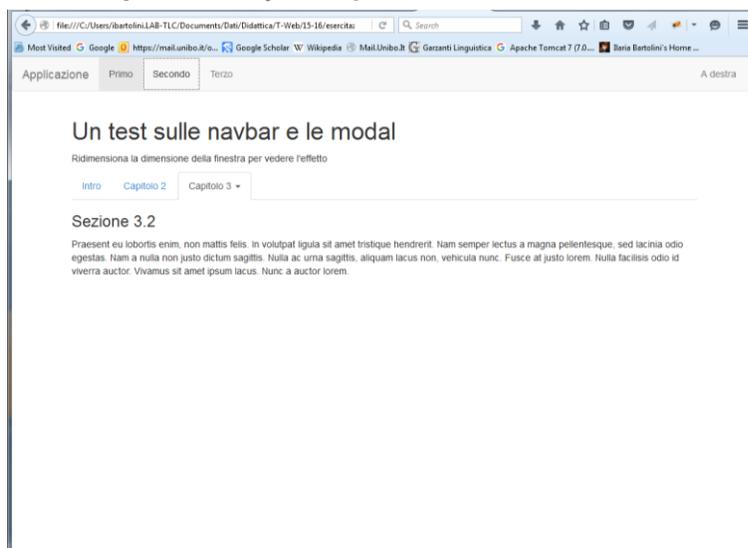
Esercitazione 1

13

13

Esempio HTML 5 CSS 3: “Navbar e modal”

“01_TecWeb.zip”, directory `esempi/HTML5CSS3/navbar.html`



Esercitazione 1

14

14

APPENDICE

(implementazione di un tunnel...)

15

Analisi dati scambiati tramite HTTP REQ/RESP via tunnel

- Un tunnel TCP è un programma che “ascolta” su una specifica porta TCP dell'host su cui viene eseguito e...
 - inoltra tutti i dati in ingresso (es: HTTP REQUEST) a un ben definito endpoint remoto (HOST+PORT)
 - restituisce tutti i dati ottenuti in risposta dall'endpoint remoto (es: HTTP RESPONSE) al richiedente iniziale



- Possiamo utilizzarlo per “monitorare” il flusso di dati (**caratteri**) che costituisce lo stream HTTP
 - basta **lanciare il tunnel sulla propria macchina...**
 - ...e **modificare adeguatamente l'URL richiamato dal browser al fine di incanalare richieste e risposte attraverso il tunnel**

16

Un esempio di tunnel TCP

- Dopo aver estratto i file presenti nell'archivio "01_TecWeb.zip" dell'esercitazione, nella directory **tunnel** trovate
 - una libreria Java **soap.jar**, contenente l'implementazione del tunnel
 - uno script di avvio **tunnel.sh** o **tunnel.bat**, che manda in esecuzione il tunnel su **localhost**
- Il tunnel richiede come parametri
 - la porta su cui porsi in ascolto sulla macchina locale
es: **8081**
 - il nome (o l'indirizzo IP) della macchina remota a cui inoltrare le richieste
es: **lia.disi.unibo.it**
la porta TCP su cui è in ascolto il server remoto che ci interessa
es: **80** (default per i server Web)
- Infine, nel browser
 - **come deve essere modificato l'URL della home page dei corsi del LIA per osservare il traffico HTTP nel tunnel?**

Richieste e risposte HTTP

L'interfaccia grafica del tunnel mostra il contenuto delle HTTP REQUEST e HTTP RESPONSE scambiate tra browser e server

- **quante e quali parti studiate nella teoria riuscite a riconoscere?**
- **perché non una sola coppia di REQ+RESP, ma tante in successione?**
- **riuscite a individuare le coppie corrispondenti?**

The screenshot shows a window titled "TCP Tunnel/Monitor: Tunneling localhost:8081 to lia.disi.unibo.it:80". It is split into two panes. The left pane, titled "From localhost:8081", shows a sequence of HTTP requests: a GET for "/Courses/" and two GETs for "/Courses/Top.html" and "/Courses/Bottom.html". The right pane, titled "From lia.disi.unibo.it:80", shows the corresponding HTTP responses, including status "200 OK", headers like "Date", "Server", "ETag", "Accept-Ranges", "Content-Length", "Keep-Alive", "Connection", and "Content-Type", followed by HTML content including a meta tag and a title.