



Alma Mater Studiorum – Università di Bologna

Scuola di Ingegneria

Tecnologie Web T
A.A. 2024 – 2025

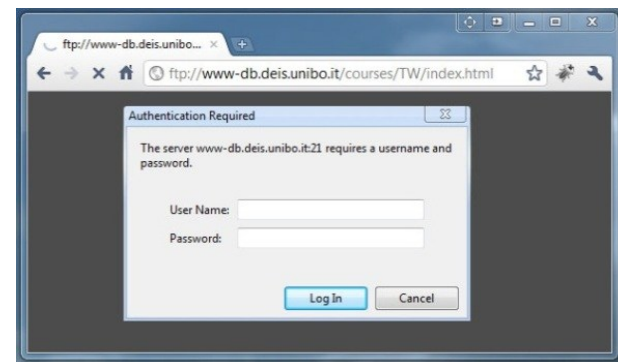
Esercitazione 1
URI, HTTP, HTML, CSS

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

URL... non solo pagine Web

- Accesso a una risorsa via HTTP
 - <http://lia.disi.unibo.it/Courses/twt2324-info/>
- Scaricamento della stessa risorsa via FTP, almeno tradizionalm
(Chrome version <= 88 || Firefox version <= 90)
 - <ftp://lia.disi.unibo.it/Courses/twt2324-info/> ????
 - <ftp://ftp.iinet.net.au/debian/debian-cd/>
- Streaming di file multimediali
(possibili client: VLC, Windows Media Player, ...)
 - <mms://151.1.245.36/rtl102.5lq>
- 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

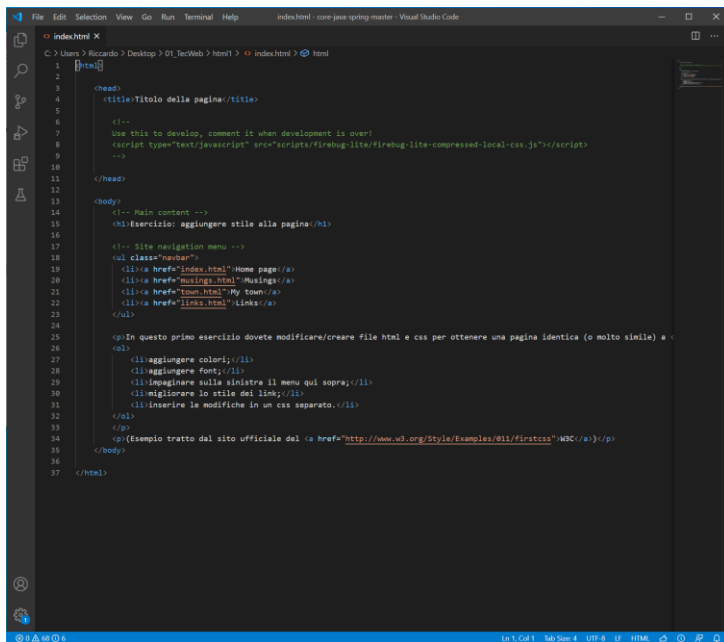
Ispezione di codice HTML esistente

- 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
 - Eclipse IDE
 - ...

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*, ...)



```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <title>Titolo della pagina</title>
6
7     <!--
8      Use this to develop, comment it when development is over!
9      <script type="text/javascript" src="scripts/firebug-lite/firebug-lite-compressed-local-css.js"></script>
10    -->
11   </head>
12
13   <body>
14     <!-- Main content -->
15     <h1>Esercizio: aggiungere stile alla pagina</h1>
16
17     <!-- Site navigation menu -->
18     <ul class="menu">
19       <li><a href="index.html">Home page</a>
20       <li><a href="musings.html">Musings</a>
21       <li><a href="town.html">My town</a>
22       <li><a href="links.html">Links</a>
23     </ul>
24
25     <!-- In questo primo esercizio dovete modificare/creare file html e css per ottenere una pagina identica (o molto simile) a
26     <ul>
27       <li><a href="#">aggiungere colori</a>
28       <li><a href="#">aggiungere font</a>
29       <li><a href="#">impaginare sulla sinistra il menu qui sopra</a>
30       <li><a href="#">migliorare lo stile dei link</a>
31       <li><a href="#">inserire le modifiche in un css separato</a>
32     </ul>
33   </body>
34   <!--(Esempio tratto dal sito ufficiale del <a href="http://www.w3.org/Style/Examples/011/firstcss">W3C</a>)
35 </html>
```

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.

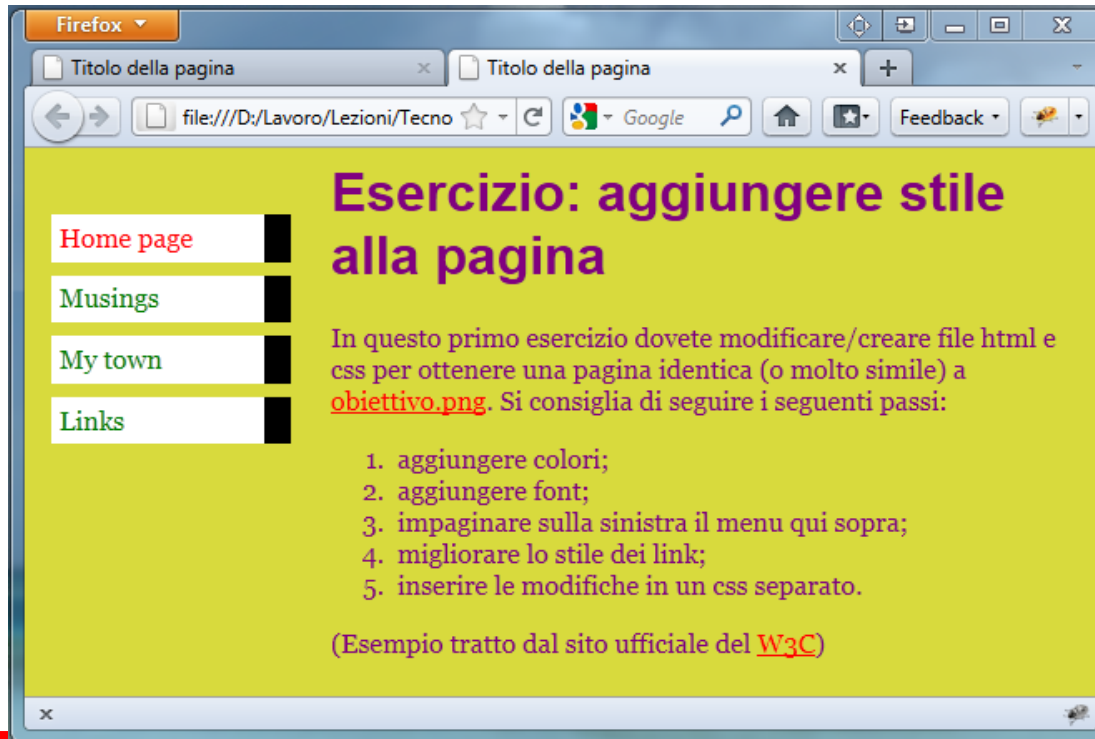
(Esempio tratto dal sito ufficiale del [W3C](#))

- 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



Ottenere il seguente risultato:

- ## Un semplice esempio di pagina HTML

una lista	paragrafi e immagini	un iframe contenente una tabella
-----------	----------------------	----------------------------------

[illegible]

Questo è un iframe :

	A	B	C	D
1	cella A1	cella B1	cella C1	cella D1
2	cella A2	contenuto presente all'interno della cella B2		contenuto presente all'interno della cella D2
3	cella A3	cella B3	cella C3	

Una form HTML richiede un pulsante di invio!

APPENDICE

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

Esempio “Lisa Simpson”: formattazione tipografica

“01_TecWeb.zip”, directory *esempi/Simpson*



Esempio “PosizionamentoSenzaFrame”: layout liquidi

“01_TecWeb.zip”, directory “*esempi/PosizionamentoSenzaFrame*”



Esempio HTML 5 CSS 3: “Le griglie”

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



Esempio HTML 5 CSS 3: “Navbar e modal”

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

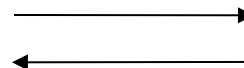
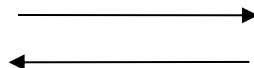


APPENDICE

(implementazione di un tunnel...)

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**

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?

```
TCP Tunnel/Monitor: Tunneling localhost:8081 to lia.deis.unibo.it:80

From localhost:8081
GET /Courses/ HTTP/1.1
Host: localhost:8081
Connection: keep-alive
Referer: http://localhost:8081/
Cache-Control: max-age=0
Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.30.2 (KHTML, like Gecko) Chrome/26.0.1410.43 Safari/534.30
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

GET /Courses/Top.html HTTP/1.1
Host: localhost:8081
Connection: keep-alive
Referer: http://localhost:8081/Courses/
Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.30.2 (KHTML, like Gecko) Chrome/26.0.1410.43 Safari/534.30
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

GET /Courses/Bottom.html HTTP/1.1
Host: localhost:8081
Connection: keep-alive
Referer: http://localhost:8081/Courses/
Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.30.2 (KHTML, like Gecko) Chrome/26.0.1410.43 Safari/534.30
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

From lia.deis.unibo.it:80
HTTP/1.1 200 OK
Date: Tue, 08 Mar 2011 09:42:41 GMT
Server: Apache/2.2.3 (Unix) mod_jk/1.2.19
Last-Modified: Thu, 02 Sep 2010 11:06:55 GMT
ETag: "dc68-186-cfe0b1c0"
Accept-Ranges: bytes
Content-Length: 390
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<html>

<head>

<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">

<title>Didattica al LIA</title>

</head>
```