



PROGRAMMAZIONE WEB HTTP E HTML

Prof. Ada Bagozi
ada.bagozi@unibs.it



Architetture client-server



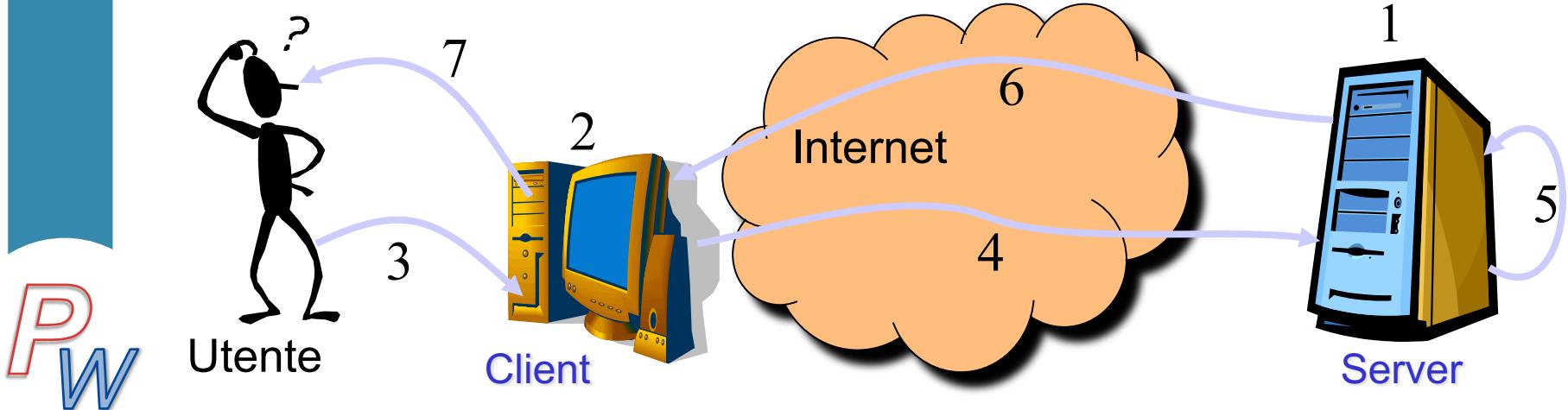
Le applicazioni Web sono organizzate secondo la cosiddetta *architettura client-server*:

- ✓ se l'utente opera con un calcolatore connesso a una rete (o comunque un dispositivo capace di connettività, come un telefono cellulare), l'unico sottosistema che deve essere eseguito localmente è l'*interfaccia utente*, chiamato in questo caso componente *client* dell'applicazione
- ✓ i sottosistemi di *gestione dati* e di *logica applicativa*, che costituiscono il componente *server* dell'applicazione, possono risiedere ed essere eseguiti su un calcolatore remoto (cioè fisicamente distante), purché appunto connesso al dispositivo impiegato dall'utente

Comunicazione tra client e server



1. il componente **server** (*server web*) viene messo in esecuzione sul calcolatore server dal gestore del servizio (www.curioso.it), e si pone in ascolto sulla rete
2. il componente **client** (*browser*) viene eseguito dall'utente
3. l'utente interagisce con il client per elaborare la **richiesta** (di un file, che chiameremo *risorsa*)
4. il client invia la **richiesta** al server
5. ricevendo la **richiesta**, il server si attiva per produrre una **risposta** (che conterrà la *risorsa*)
6. il server manda la **risposta** al client attraverso la rete
7. ricevendo la **risposta**, il client presenta il contenuto della *risorsa* all'utente



Comunicazione tra client e server



Il *client* invia la richiesta al *server* in una forma simile a questa

```
GET /CoseInteressanti/BelloQuesto.txt HTTP/1.1  
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5;  
Windows NT 5.0) Accept: */*
```

Ricevendo la richiesta il *server* cerca il file richiesto (*/CoseInteressanti/BelloQuesto.txt*) e, trovatolo, produce un messaggio del tipo:

```
HTTP/1.1 200 OK  
Server: Microsoft-IIS/5.0  
Content-Type: text/plain  
Content-Length: 14
```

Saluti a tutti

HTTP: un protocollo applicativo



Perché un client e un server possano comunicare:

- ✓ non è sufficiente che siano in grado di scambiarsi richieste e risposte
- ✓ è necessario che tali richieste e risposte siano scritte in un **formato concordato**

Client e server devono “parlare la stessa lingua”, cioè devono condividere uno stesso **protocollo applicativo**.

Nel caso del Web il protocollo applicativo, che specifica il formato dell'intestazione sia della richiesta del browser sia della risposta del server web, è **HyperText Transfer Protocol (HTTP)**

Rivediamo l'esempio di risposta:

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Type: text/plain
Content-Length: 14
```

Saluti a tutti

} Intestazione HTTP, con dati sul server web
che ha prodotto la risposta e sul file restituito

Linea bianca: separatore tra intestazione e corpo

Corpo del messaggio: contenuto del file restituito

Identificazione delle risorse

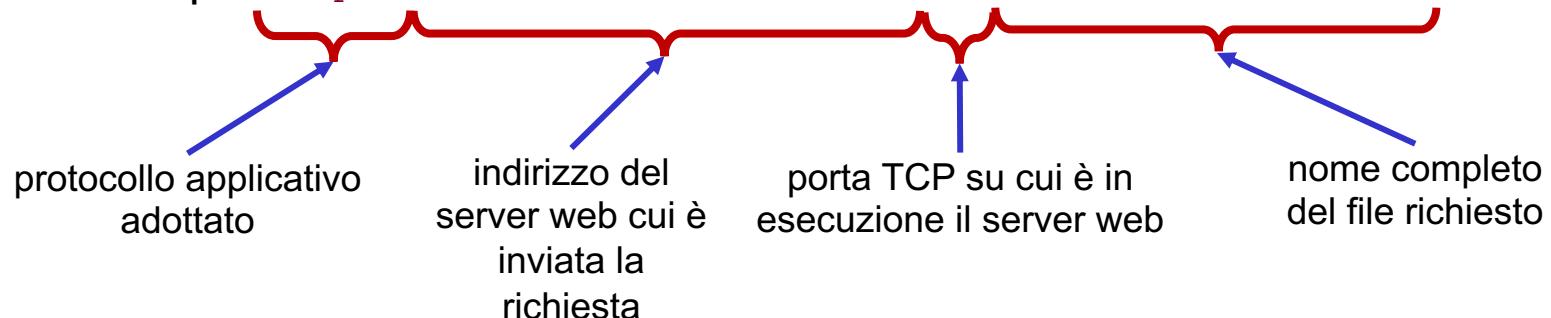


Per poter accedere alle risorse messe a disposizione dai server sulla rete, è necessario che ogni risorsa risulti *univocamente identificabile* da parte dei client

Nel caso di *Internet* si è standardizzato uno schema uniforme di identificazione di ogni risorsa presente sulla rete, che consente di assegnare un metodo di accesso e un indirizzo, chiamati congiuntamente *Uniform Resource Locator (URL)*

In generale, un *URL* ha la forma **protocollo://host:portaTCP/risorsa**

Per esempio `http://server1.isttec.liuc.it:80/dida/calendario.html`



Che cos'è una risorsa?



Una richiesta (HTTP) coincide sempre con l'invocazione di una risorsa:

- un *file HTML*, ovvero un file di testo opportunamente codificato per essere letto da un web browser
- un *foglio di stile (CSS)*, ovvero un file di testo opportunamente codificato per dare uno stile grafico alle pagine HTML
- *contenuti multimediali* quali immagini, filmati, file audio
- *componenti* da scaricare per eseguirli sul client (come Java Applet, ActiveX components, ...)
- *componenti* da eseguire sul server (come CGI, Java Servlet, ...)
- *script eseguiti lato server* che generano dinamicamente la pagina HTML e che sono codificati in opportuni linguaggi (PHP, Java, ...)
- *script eseguiti lato client* e che sono codificati in opportuni linguaggi (e.g., Javascript)
- altri *file da scaricare* dal server (JSON, XML, TXT, ZIP, PDF ...)

HyperText Markup Language



- ✓ HTML (**HyperText Markup Language**) è un linguaggio di markup per gli ipertesti
- ✓ Permette di indicare come disporre gli elementi (testo, immagini, tavole, ecc.) all'interno di una pagina
- ✓ Le indicazioni vengono date attraverso degli appositi marcatori, detti **tag**
- ✓ Un **file HTML** è un comune file di testo ASCII che viene interpretato dal **Web browser** per presentare le informazioni in una pagina Web
- ✓ Attualmente, è disponibile la versione **HTML5**

Tag HTML



Ogni **tag** è formato da un nome, che lo identifica, racchiuso da «brackets»

<tag>

Solitamente gli elementi da visualizzare sono racchiusi tra un tag di apertura e uno di chiusura, che presenta il carattere “ / ” (slash) anteposto al nome del tag

<tag> ... </tag>

Esiste anche la forma contratta, nel caso in cui non ci sia contenuto tra tag di apertura e tag di chiusura

<tag/>

Annidamento dei tag HTML



I tag possono essere **annidati** a più livelli
(!!! attenzione all'ordine di annidamento !!!)

Annidamento corretto

```
<tag1>  
<tag2>  
...  
</tag2>  
</tag1>
```

Annidamento errato!!!

```
<tag1>  
<tag2>  
...  
</tag1>  
</tag2>
```

Attributi dei tag HTML



Un tag può avere uno o più **attributi** che forniscono ulteriori informazioni da presentare

Gli attributi sono espressi nella forma

attributo="valore"

Gli attributi vengono inseriti nel tag di apertura dopo il nome del tag stesso

```
<tag attributo1="valore" attributo2="valore">  
...  
</tag>
```

Struttura di un documento HTML



Un documento HTML è un file ASCII racchiuso fra i tag `<html>` e `</html>`

Presenta un'intestazione tra i tag `<head>` e `</head>`

Presenta un corpo tra i tag `<body>` e `</body>`

L'intestazione contiene informazioni sul documento

Nel corpo troviamo il contenuto del documento e i tag per la resa visiva

```
<html>
  <head>
<title> ... </title>
  </head>

  <body>
...
  </body>
</html>
```

Classificazione dei tag HTML (I)



Flusso di tipo inline – L'inserimento dei tag non provoca la creazione di un nuovo paragrafo

```
<tt>, <i>, <b>, <big>, <small>, <em>, <strong>, <dfn>,  
<code>, <samp>, <kbd>, <var>, <cite>, <abbr>,  
<acronym>, <a>, <img>, <object>, <br>, <script>,  
<map>, <q>, <sub>, <sup>, <span>, <bdo>, <input>,  
<select>, <textarea>, <label>, <button>
```

Flusso di tipo blocco - L'inserimento dei tag provoca la creazione di un nuovo paragrafo

```
<p>, <h1>...<h6>, <ol>, <ul>, <pre>, <dl>, <div>,  
<noscript>, <blockquote>, <form>, <hr>, <table>,  
<fieldset>, <address>
```

Classificazione dei tag HTML (II)



Tag per controllare il flusso del testo nel documento: `<p>`, `
`,
`<h1>..<h6>`, `<div>`, ``

Tag per la formattazione di base: `<tt>`, `<i>`, ``, `<big>`, `<small>`, `<sub>`,
`<sup>`

Tag per la formattazione semantica e le revisioni: ``, ``,
`<dfn>`, `<code>`, `<samp>`, `<kbd>`, `<var>`, `<cite>`, `<abbr>`, `<acronym>`,
`<blockquote>`, `<q>`, `<pre>`, `<ins>`, ``, `<address>`

Tag per la rappresentazione di liste: ``, ``, ``, `<dl>`, `<dt>`, `<dd>`

Tag per la rappresentazione di tabelle: `<table>`, `<caption>`, `<tr>`, `<td>`,
`<th>`, `<thead>`, `<tbody>`, `<tfoot>`, `<col>`, `<colgroup>`

Tag per la rappresentazione di collegamenti: `<a>`, `<link>`, `<base>`

Tag per l'inclusione di oggetti multimediali: ``, `<map>`, `<area>`,
`<object>`

Tag per l'interazione con l'utente: `<form>`, `<input>`, `<textarea>`,
`<button>`...

Attributi HTML standard



id: ID unico (utilizzato per riferirsi all'elemento negli script)

class: lista di classi (utilizzati per attribuire uno o più stili globali all'elemento; la lista è delimitata da spazi)

style: informazioni di stile (utilizzato per fornire uno stile specifico all'elemento)

title: titolo informativo (molti browser lo renderizzano come tooltip dell'elemento)

lang: codice lingua (codici linguistici come da standard I18N, ad es. "it" o "en-us")

dir: direzione di scrittura (rtl, destra-a-sinistra, o ltr, sinistra-a-destra)

onclick, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**,
onmousemove, **onmouseout**, **onkeypress**, **onkeydown**, **onkeyup**:
gestori eventi (utilizzati per associare degli script agli eventi corrispondenti)

Tag HTML – Le tavole (I)



- ✓ Le tavole HTML sono uno strumento per disporre informazioni in righe e colonne
- ✓ Le tavole non sono progettate per creare layout di pagina, ma solo per strutturare informazioni in forma tabulare; utilizzare le tavole per creare layout rende questi ultimi poco portabili ed è fortemente sconsigliato
- ✓ Le tavole fanno parte degli elementi di tipo blocco, quindi possono apparire direttamente nel **<body>** di un documento o in un contenitore **<div>**, e non devono mai essere nidificate in elementi come **<p>**
- ✓ L'elemento base della definizione delle tavole è **<table>**

Tag HTML – Le tavole (II)



Le larghezze delle colonne e della tabella stessa (attributo **width**) si possono specificare:

- ✓ in pixel (**width="10"**)
- ✓ in percentuale, rispetto allo spazio disponibile per la tabella (**width="10%"**)

Solo per le colonne, si possono usare anche i seguenti sistemi:

- ✓ proporzionalmente, rispetto alla dimensione richiesta dalla tabella (**width="10*"**)
- ✓ per richiedere il minimo spazio necessario, si può usare la forma **width="0*"**

Se non si specifica una larghezza per una tabella, lo spazio è quello necessario a dare larghezza minima a tutte le colonne

Tag HTML – Le tabelle (esempio)



Sezionamento Verticale

Categoria	Preferenza		
	Pianura	Collina	Montagna
Maschi	54%	24%	22%
Femmine	63%	21%	16%

Dati ottenuti mediante campionamento pseudo-casuale



Tag HTML – Le tavole (esempio)



```
<table border="1" cellpadding="25" cellspacing="0" rules="groups">
<caption><h2>Sezionamento Verticale</h2></caption>
<colgroup span="2" />
<colgroup>
  <col style="background-color:lightgreen" width="20%" />
  <col style="background-color:yellow" width="20%" />
  <col style="background-color:lightpink" width="20%" />
</colgroup>
<thead style="background-color:white">
  <tr>
    <th rowspan="2" colspan="2" > </th> <th colspan="3" >Preferenza</th>
  </tr>
  <tr>
    <th>Pianura</th> <th>Collina</th> <th>Montagna</th>
  </tr>
</thead>
<tfoot>
  <tr align="center">
    <td colspan="5" >Dati ottenuti mediante campionamento pseudo-casuale</td>
  </tr>
</tfoot>
<tbody>
  <tr align="center">
    <th rowspan="2" >Categoria</th>
    <th>Maschi</th> <td>54%</td> <td>24%</td> <td>22%</td>
  </tr>
  <tr align="center">
    <th>Femmine</th> <td>63%</td> <td>21%</td> <td>16%</td>
  </tr>
</tbody>
</table>
```

Tag HTML – Le immagini



****: inclusione di un'immagine

Contenuto: vuoto

Attributi: HTML standard, **src**, **alt**, **longdesc**, **width**, **height**, **usemap**

- ✓ Inserisce nel documento l'immagine esterna referenziata dall'URL nell'attributo **src**
- ✓ Il testo alternativo per l'immagine (**alt**) è una caratteristica essenziale per un documento HTML ad alta accessibilità
- ✓ L'attributo **longdesc** può essere usato per puntare all'URL di un documento che descrive nel dettaglio l'immagine
- ✓ Gli attributi **width** e **height** dovrebbero essere sempre usati per fornire al browser un suggerimento sulle dimensioni da riservare per l'immagine sulla pagina. Se non corrispondono alle dimensioni reali dell'immagine, questa verrà ridimensionata di conseguenza (in maniera proporzionale se si specifica solo uno degli attributi)

Tag HTML – I link ipertestuali



```
<a href="destinazione" target=?>contenuto del link</a>
```

- ✓ È il tag che identifica i link, ossia gli elementi che, se cliccati, rimandano ad un punto diverso all'interno dello stesso (`href="#bookmark"`) o di un altro documento (`href="http://www..."`), permettono di mandare un'email (`href="mailto:..."`), permettono di scaricare un file (per es., `href="fileName.zip"`)
- ✓ Se il link rimanda ad un punto particolare all'interno del documento, quel punto dovrà essere univocamente identificato all'interno del documento stesso tramite l'attributo `id` (per es., `<h2 id="bookmark">`)
- ✓ È possibile specificare la modalità con cui il browser deve aprire la destinazione del link (`_blank`, `_self`, che è anche il default, `_parent`, `_top`, `"framename"`)
- ✓ Il contenuto del link può essere in generale una risorsa, ma non si possono avere link nidificati

Tag HTML – Le form (I)



- ✓ I **moduli** o **form** sono tag HTML che permettono all'utente di interagire con la pagina web

```
<form name="mioForm" action=".." method="get/post">  
...  
</form>
```

- ✓ L'attributo **name** indica il nome del form, mentre l'attributo **action** indica l'azione da compiere, la pagina da richiamare, lo script da eseguire, ecc. (in altri termini, la risorsa da richiedere)
- ✓ L'attributo **method** (*get* o *post*) permette di specificare il metodo HTTP per l'invio della richiesta
- ✓ I campi della form sono utilizzati per inviare al server dei parametri di input

Tag HTML – Le form (II)



<input>

Contenuto: vuoto

Attributi: HTML standard, **type**, **name**, **value**, **size**, **maxlength**,
checked, **disabled**, **readonly**

L'elemento **<input>** viene usato per generare gran parte dei campi all'interno dei moduli; la chiave della sua versatilità è l'attributo **type**, che può assumere i seguenti valori:

- ✓ **text**: crea una riga di input testuale
- ✓ **password**: come text, ma maschera i caratteri digitati
- ✓ **checkbox**: crea una casella di controllo
- ✓ **radio**: crea un pulsante di opzione
- ✓ **submit**: crea un bottone per l'invio del modulo
- ✓ **reset**: crea un bottone per la reinizializzazione del modulo
- ✓ **file**: crea un controllo per l'upload di un file
- ✓ **hidden**: crea un campo nascosto nel modulo
- ✓ **button**: crea un bottone

Tag HTML – Le form (III)



<input>

- ✓ L'attributo **value** fornisce:
 - ✓ la stringa di inizializzazione per i tipi *text*, *password*, *hidden*, *file*
 - ✓ la label per i controlli di tipo *submit*, *reset* e *button*
- ✓ L'attributo **size** fornisce la larghezza del campo, espressa in pixel o in caratteri per i tipi *text* e *password*
- ✓ L'attributo **maxlength** fornisce il massimo numero di caratteri digitabili nei campi di tipo *text* e *password*
- ✓ L'attributo booleano **checked** determina se i campi di tipo *checkbox* e *radio* saranno inizialmente selezionati
- ✓ Gli attributi booleani **disabled** e **readonly** possono essere utilizzati per disabilitare e/o rendere di sola lettura i campi

Il tag <object>



Introdotto in HTML4 per includere un oggetto generico nella pagina HTML (per esempio, file multimediali audio/video, Java applets, oggetti ActiveX, Flash)

- ✓ Presenta come attributi specifici:
 - ✓ **data**, un URL che identifica il file da caricare (es. immagini, video, audio)
 - ✓ **type**, indica il tipo di contenuto del file (MIME type)

Il tag <object>



```
<!DOCTYPE html>
<html lang="en">
<head>
    ...
</head>
<body>

    <h1>Video Incorporato</h1>

    <object data="mio_video.mp4" type="video/mp4" width="640"
           height="360">
        <p>Il tuo browser non supporta il tag video.
            Puoi scaricare il video <a href="mio_video.mp4">qui</a>.
        </p>
    </object>

</body>
</html>
```

Comprendere i MIME types



MIME (Multipurpose Internet Mail Extension) è uno standard per indicare il tipo di contenuto di un file, a supporto dei browser

- ✓ Struttura tipo/specifica, ad indicare la categoria generale e dettagli più specifici sul tipo di dati
 - ✓ `text/plain` o `text/html`, `image/jpeg` o `image/png` ,
`audio/mp3` o `audio/wav` , `video/mp4` o `video/webm` ,
`application/pdf` o `application/json`
- ✓ Pensati per assicurare compatibilità e interoperabilità (interpretazione corretta dei dati) e sicurezza (previene l'esecuzione di file dannosi)

HTML5 – Nuovi tag (I)



Vengono introdotti elementi specifici per contenuti multimediali (tag `<video>` e `<audio>`), in sostituzione di software di terze parti (e.g., Adobe FlashPlayer, QuickTime)

- ✓ Vengono estesi a tutti i tag alcuni attributi, in particolare quelli relativi all'accessibilità; maggiore attenzione alla responsività delle pagine Web
- ✓ Supporto dell'elemento Canvas per utilizzare Javascript al fine di creare animazioni e grafica bitmap
- ✓ Introduzione della geolocalizzazione, soprattutto per la diffusione dei sistemi operativi mobili Android e IOS
- ✓ I cookie vengono sostituiti da Web Storage, più efficiente; meccanismo di caching
- ✓ Vengono introdotti altri raffinamenti, come regole più stringenti sulla strutturazione del testo e alcuni controlli aggiuntivi, mentre vengono deprecati alcuni elementi scarsamente utilizzati (e.g., `<applet>` a favore di `<object>`) o eliminati elementi considerati dannosi per l'accessibilità (e.g., `<frame>`, `<frameset>`, `<noframes>`)

HTML5 – Nuovi tag (II)



Vengono introdotti nuovi tag in ottica accessibilità

- <header>
- <nav>
- <section>
- <article>
- <aside>
- <figure>
- <details>
- <footer>



Web dinamico



- ✓ Gli attori dell'architettura client-server dinamico
 - ✓ Web browser dinamico, esegue programmi lato client, sviluppati usando client-side scripting (Javascript, VisualBasic scripting) o come client-side components (Java Applets, Microsoft ActiveX, Macromedia Flash movies)
 - ✓ Web server dinamico, esegue applicazioni server-side, tecnologie CGI, Java Servlet e server-side scripting (e.g., JSP, PHP)
- ✓ Pensato per produrre pagine “*on-the-fly*” in base alle esigenze dell’utente e a contenuti strutturati (per ex. database)
 - ✓ per esempio, le news sulla pagina Web di un giornale on-line vanno aggiornate ogni giorno pescando le nuove notizie da un repository sottostante
 - ✓ la pagina HTML di risposta “*non esiste*” così com’è sul server, viene generata dinamicamente in base alle elaborazioni effettuate

Client-side scripting



Il codice script è interpretato dal browser

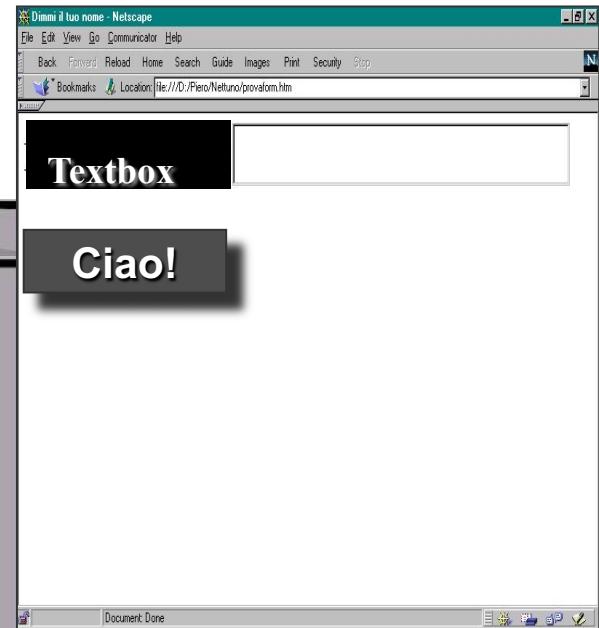
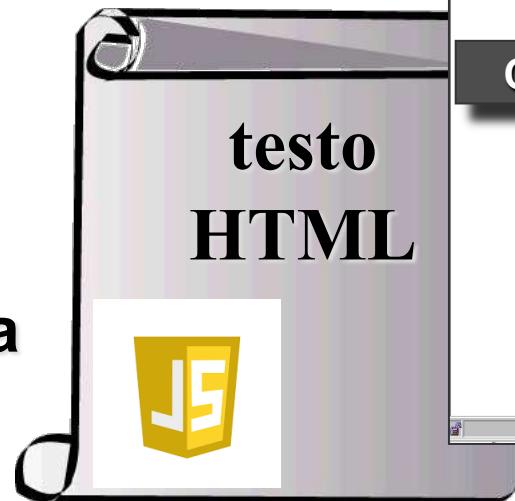
Contenuto

+

script

=

pagina interattiva



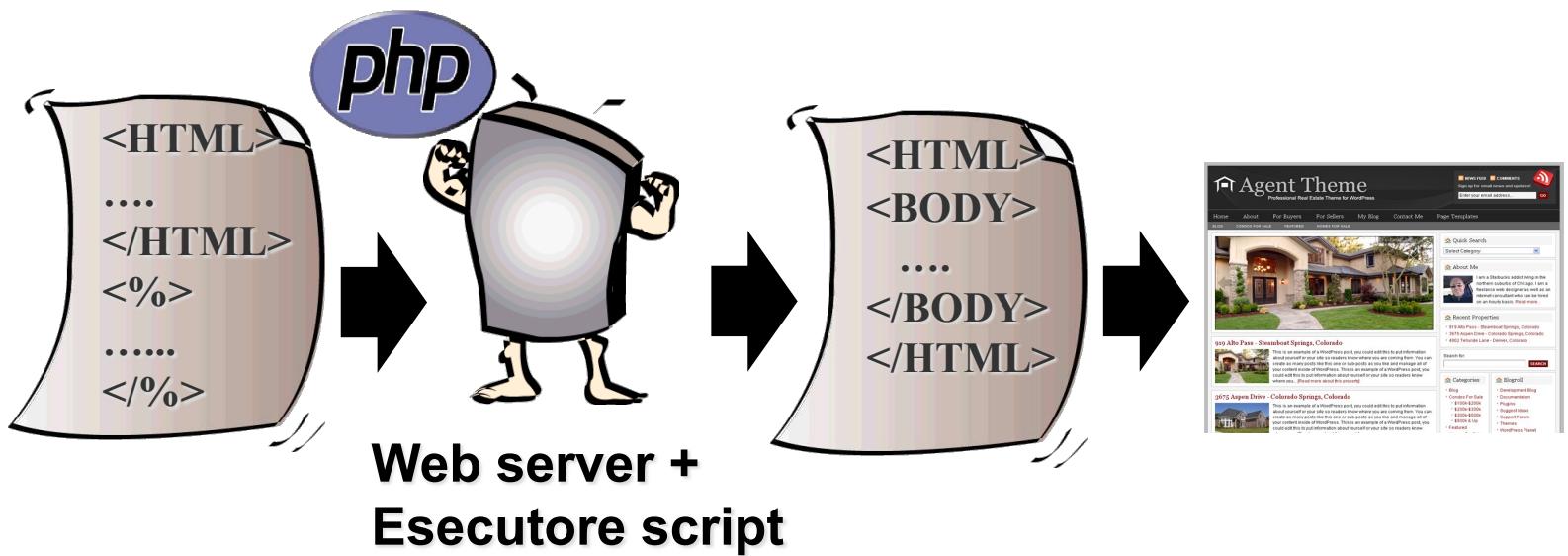
Server-side scripting



Vengono inserite istruzioni per la generazione dei contenuti dinamici all'interno della pagina HTML

Il codice è interpretato dal server

Il browser riceve HTML puro



Architetture multi-tier (I)



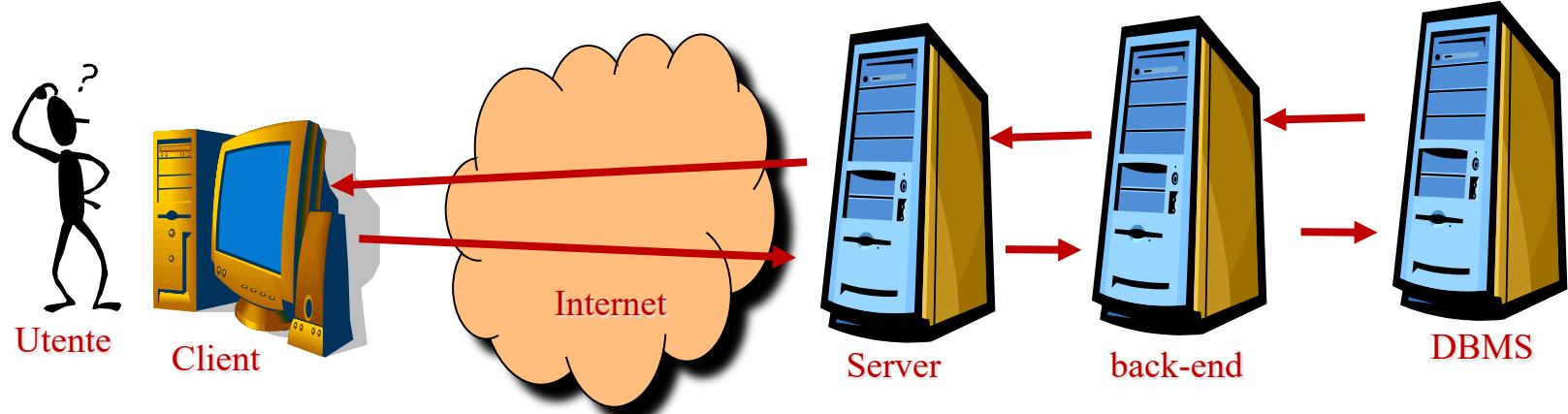
Combinazione di due sottosistemi *client-server*:

- ✓ il browser opera come *client* verso il *server web*, inviandogli una richiesta e aspettando da esso una risposta
- ✓ il *server web* assume temporaneamente un ruolo di *client*, inviando una richiesta a un *altro server* (il programma di *back end* o il *database*) e attendendo da esso una risposta

Architettura basata su un principio di *deleghe successive*:

- ✓ un *client* non è in grado di svolgere un compito applicativo e quindi si rivolge a un *server*, che può a sua volta trasformarsi in un *client* di un ulteriore *server* se non è in grado di soddisfare la richiesta

Architetture multi-tier (II)



Svantaggio

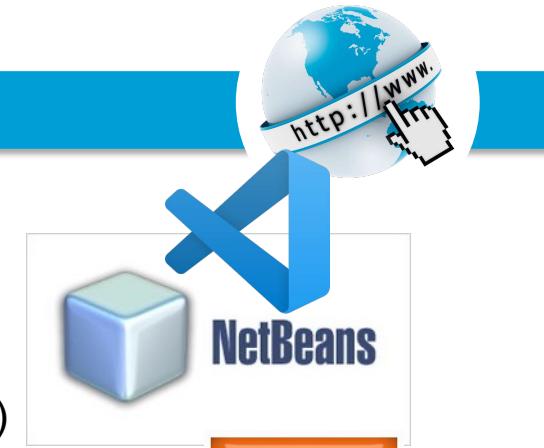
- complessità architetturale

Vantaggio

- specializzazione: ogni sottosistema è realizzato per svolgere un compito particolare e deve rispettare il protocollo applicativo con i sottosistemi delegati

Gli strumenti per iniziare

- Un ambiente di sviluppo o IDE
- Un server Web
 - XAMPP (Apache, MySQL/MariaDB, PHP, Perl)
 - MAMP (per macOS e Windows)
 - Laragon (ambiente di sviluppo leggero per PHP)
 - LAMP (Linux, Apache, MySQL, PHP)
 - PHP Built-in Server (php -S localhost:8000)
- Un browser
- Database relazionale: MySQL
- Gestore DB: phpMyAdmin, Adminer, MySQL Workbench, Sequel Pro (mac), DBeaver



Alcuni link utili ...



- ✓ Tutorial della W3C su HTML
(<http://www.w3schools.com/html/default.asp>)
- ✓ Specifica HTML 4 (<http://www.w3.org/TR/html401/>)
- ✓ Specifica HTML 5 ([http://www.w3schools.com/html/html5_intro.asp/](http://www.w3schools.com/html/html5_intro.asp))
- ✓ E molto altro ancora in rete



PROGRAMMAZIONE WEB

HTTP E HTML





PROGRAMMAZIONE WEB

HTML E FOGLI DI STILE

Prof. Ada Bagozi
ada.bagozi@unibs.it



Cascading Style Sheets (CSS)



Strumento che permette di separare il contenuto di una pagina Web dalla sua impaginazione grafica

File di testo che definisce delle **regole di formattazione** da applicare alla pagina HTML

Viene applicato ad un file HTML

Sintassi delle regole CSS



Tre parti fondamentali: un *selettore*, una *proprietà* e un *valore*

```
selettore {proprietà: valore}
```

- ✓ Selettore: l'elemento (tag) HTML di cui definire lo stile; è possibile raggruppare più selettori, separandoli con una virgola (es. `h1, h2, h3 {color: green}`)
- ✓ Proprietà: l'attributo che si desidera modificare; se si vogliono modificare più proprietà, vanno inserite separate da ';' (es. `{text-align:center; color:red}`)
- ✓ Valore: il valore che deve assumere l'attributo modificato; se il valore è costituito da più parole, queste vanno messe tra virgolette e separate da uno spazio (es. `{font: "sans serif"}`)

Regole CSS: il selettore



- ✓ Con il selettore possiamo indicare:
 - ✓ uno o più elementi (tag) HTML (**type selector**)
`h1 { ... } h1, h2 { ... }`
 - ✓ tutti gli elementi HTML (**universal selector**)
`* { ... }`
 - ✓ tutti gli elementi discendenti di un altro elemento (**descendant selector**)
`h1 p { ... }` seleziona tutti i **p** discendenti di **h1**
 - ✓ tutti gli elementi figli di un altro elemento (**child selector**)
`h1 > p { ... }` seleziona tutti i **p** figli di **h1** (1° livello!!!)
 - ✓ tutti gli elementi il cui attributo class è **value** (**class selector**)
`.value { ... }` seleziona tutti gli elementi con
`class="value"`
`h1.value { ... }` seleziona tutti gli **h1** con `class="value"`
 - ✓ l'elemento il cui attributo id è **value** (**id selector**)
`#value { ... }` (nel file HTML esiste un solo tag il cui id è **value**!)

CSS ed ereditarietà

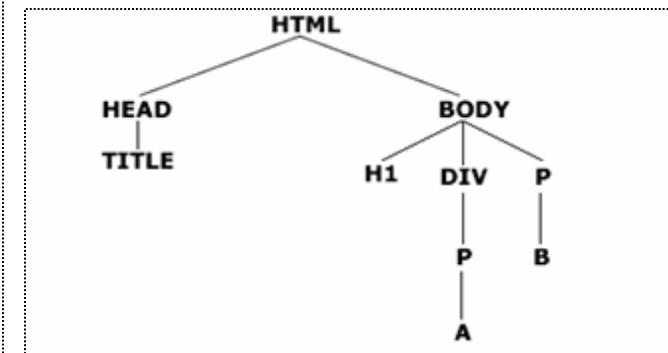


Meccanismo fondamentale dei CSS è l'**ereditarietà**: molte proprietà impostate per un elemento sono automaticamente ereditate dai suoi discendenti

```
<html>
  <head>
    <title>Struttura del documento</title>
  </head>

  <body>
    <h1>Titolo</h1>
    <div>
      <p>Primo
        <a href="pagina.htm">paragrafo</a>
      </p>
    </div>
    <p>Secondo<b>paragrafo</b></p>
  </body>
</html>
```

Un elemento si dice **genitore** quando contiene altri elementi.
Un elemento si dice **figlio** quando è racchiuso in un altro elemento.



BODY è **genitore (parent)** di H1, DIV e P

H1 è **figlio (child)** di BODY

BODY è un **antenato (ancestor)** di B

B è un **discendente (descendant)** di BODY

Collegare HTML e CSS



Per collegare il file CSS ad un file HTML è necessario utilizzare il tag **<link>** nell'intestazione della pagina Web

```
<head>
  <link rel="stylesheet" type="text/css"
        href="mystyle.css"/>
</head>
```

- ✓ **rel**: attributo obbligatorio, descrive il tipo di relazione tra il documento e il file collegato
- ✓ **href**: attributo obbligatorio, serve a definire l'URL assoluto o relativo al foglio di stile
- ✓ **type**: attributo opzionale, identifica il tipo MIME del file da collegare: per i CSS l'unico valore possibile è **text/css**
- ✓ **media**: attributo opzionale, identifica il supporto (screen, print, etc) cui applicare il foglio di stile

CSS – Esempio (I)



HTML

```
<html>
<head>
    <title>Esempio dell'uso dei CSS</title>
    <link href="esempio.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <h1>Titolo</h1>
    <h2>Primo paragrafo</h2>
    <div id="primo">Lorem ipsum...</div>

    <h2>Secondo paragrafo</h2>
    <div><p>Lorem ipsum...</p>
    <ul>
        <li>primo item</li>
        <li>secondo item</li>
    </ul>
    </div>

    <div>
        <table><tr><td>Prima cella</td><td>Seconda cella</td></tr>
            <tr><td>Terza cella</td><td>Quarta cella</td></tr>
        </table>
    </div>
</body>
</html>
```



CSS – Esempio (I)



Titolo

Primo paragrafo

 Lorem ipsum dolor sit amet, *consectetur adipisicing elit*, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Secondo paragrafo

 Lorem ipsum dolor sit amet, **consectetur adipisicing elit**, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

- primo item
- secondo item

Prima cella Seconda cella

Terza cella Quarta cella

CSS – Esempio (I)



```
body{  
    background: #FFE4B5;  
    font: 10pt Verdana;  
}  
  
h1{  
    color: blue;  
    font: bold 20pt Tahoma;  
}  
  
h2 {  
    color: #8B008B;  
    font: bold 16pt Tahoma;  
}  
  
table {  
    width: 300px;  
    height: 100px;  
    font: 8pt Verdana;  
    border-width: 1pt;  
    border-style: dotted;  
    border-color: green;  
}  
  
td {  
    border-style: dashed;  
    border-width: 1pt;  
    text-align: center;  
}  
  
div {  
    width: 500px;  
    border: 1pt solid red;  
    margin-top: 10px;  
}  
  
#primo {  
    border: 1pt dotted red;  
    text-align: justify;  
}  
  
div p {  
    padding-left: 50px;  
    border: 1pt dotted blue;  
}
```

CSS – Esempio (I)



Titolo

Primo paragrafo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Secondo paragrafo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

- primo item
- secondo item

Prima cella	Seconda cella
Terza cella	Quarta cella



CSS – Esempio (II)



```
<html>
  <head>
    <meta charset="utf-8">
    <title>Prova menù</title>
    <link rel="stylesheet" href="css/prova.css">
  </head>

  <body>
    <div id="menu">
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">My Library</a>
          <ul>
            <li><a href="#">Book's list</a></li>
            <li><a href="#">Authors' list</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </body>
</html>
```

HTML



CSS – Esempio (II)



```
div#menu {  
    width: 100%;  
    float: left;  
    background: #f3f3f3;  
}  
  
div#menu ul  
{  
    margin: 0;  
    padding: 0;  
    list-style-type: none;  
}  
  
div#menu li  
{  
    float:left;  
    padding: 10px;  
    border-right: 1px solid lightgrey;  
    color:black;  
}  
  
div#menu a  
{  
    display: block;  
    padding: 5px;  
    color: #000;  
    text-decoration: none;  
}  
  
div#menu a:hover  
{  
    background-color: darkgrey;  
}  
  
div#menu li ul  
{  
    display: none  
    padding: 10px;  
}  
  
div#menu li:hover ul  
{  
    display: block;  
    position: absolute;  
    z-index:1;  
    width:160px;  
    border:1px solid black;  
    background: white;  
}  
  
div#menu li li { border:none; width:150px; }  
  
li a.active  
{  
    background-color: lightgrey;  
}
```

CSS – Esempio (II)



Senza foglio di stile .css

- [Home](#)
- [My Library](#)
 - [Books List](#)
 - [Authors List](#)

Con foglio di stile .css



Vantaggi dei CSS



I CSS permettono la separazione del contenuto dallo stile quindi

Migliore gestione della grafica: è possibile scegliere il font desiderato, gestire l'interlinea e la spaziatura dei contenuti della pagina in modo facile ed efficiente

Modifica semplice dell'impaginazione (layout): modificando una regola di formattazione è possibile modificare l'aspetto di più pagine contemporaneamente

Accessibilità: eliminando l'uso improprio di elementi HTML per la grafica e l'impaginazione (es. tavole) si ottengono siti più facilmente navigabili dai programmi di lettura video

Nota sui laboratori



Creiamo un sito Web che consenta di visualizzare:

- informazioni relative ad una lista di libri, con la possibilità di aggiungere nuovi libri, cancellare o modificare gli esistenti; per ogni libro è mostrato il titolo e il cognome dell'autore
- informazioni relative agli autori, con la possibilità di aggiungere nuovi autori, cancellare o modificare gli esistenti; per ogni autore sono mostrati nome e cognome

Completeremo e arricchiremo questo esempio con nuove funzionalità passo passo durante il corso

Running example (I)



Home My Library

My online Library

A very simple example of a website created during the Web Programming and Digital Services course. The site lists the books I am currently reading or have read, along with the list of authors who have populated my readings and imagination. The website will continue to grow during this semester, completing itself gradually thanks to the implementation of web technologies that will be introduced in the course. Enjoy!

Sow an act, and you reap a habit; sow a habit, and you reap a character; sow a character, and you reap a destiny.

Home My Library

My on

Books List
Authors List

A very simple example of a website created during the Web Programming and Digital Services course. The site lists the books I am currently reading or have read, along with the list of authors who have populated my readings and imagination. The website will continue to grow during this semester, completing itself gradually thanks to the implementation of web technologies that will be introduced in the course. Enjoy!

Sow an act, and you reap a habit; sow a habit, and you reap a character; sow a character, and you reap a destiny.

[Indian proverb]

Running example (II)



Home My Library

My Books List

[Create new book](#)

A sapiente vel est.	Towne	Edit	Delete
Ab perferendis.	Schroeder	Edit	Delete

Home My Library

Insert new book

Title

Author

[Save](#) [Cancel](#)

Da così a così



- [Home](#)
- [My Library](#)
 - [Books List](#)
 - [Authors List](#)

My online Library

A very simple example of a website created during the Web Programming and Digital Services course. The website will continue to grow during this semester.

Sow an act, and you reap a habit; sow a habit, and you reap a

[Indian proverb]



[Home](#) [My Library](#)

My online Library

A very simple example of a website created during the Web Programming and Digital Services course. The site lists the books I am currently reading or have read, along with the list of authors who have populated my readings and imagination. The website will continue to grow during this semester, completing itself gradually thanks to the implementation of web technologies that will be introduced in the course. Enjoy!

Sow an act, and you reap a habit; sow a habit, and you reap a character; sow a character, and you reap a destiny.

[Indian proverb]



P
W

Alcuni link utili ...



- ✓ Tutorial della W3C su CSS (w3schools.com/css)
- ✓ Una risorsa eccellente per familiarizzare con le molte proprietà e sotto-proprietà CSS è la documentazione ufficiale di Mozilla Developer Network (MDN):
<https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>



PROGRAMMAZIONE WEB

HTML E FOGLI DI STILE

P
W



PROGRAMMAZIONE WEB

IL FRAMEWORK BOOTSTRAP

Prof. Ada Bagozi
ada.bagozi@unibs.it



Cos'è Bootstrap



Un framework open-source per la realizzazione di siti Web responsive, nativamente pensati anche per far fronte alla diffusione dei dispositivi mobili, usato da circa l'1% dei siti pubblicati in rete
(fonte Twitter/X, Luglio 2023)

Nasce nell'Agosto 2011 da un'idea di Mark Otto e Jacob Thornton (all'epoca sviluppatori nel team di Twitter), attualmente è attiva la versione 5.3.x

Nasce dalla necessità di standardizzare il set di widgets e aspetti di stile nello sviluppo di applicazioni Web in seno alla compagnia

Nato per essere interamente CSS-based, il progetto è evoluto includendo molti plug-in Javascript ed è altamente personalizzabile



Bootstrap – Getting Started



Per iniziare a lavorare con il framework, basta scaricare il pacchetto predefinito di file CSS e Javascript e includerlo all'interno del proprio progetto (<https://getbootstrap.com/docs/5.3/getting-started/download/>)

Sono richieste anche le librerie Javascript **jQuery.js**, e **Popper.js**, non incluse nella distribuzione

```
<script src="https://code.jquery.com/jquery-3.6.0.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>
```

Bootstrap – Elementi principali



CSS – Configurazioni CSS di base, definizione dello stile dei principali elementi HTML, definizione della griglia utile per impostare il layout della pagina

Componenti – Un set di widget riusabili per includere nella pagina Web iconografie, menù a tendina, barre di navigazione, finestre alert, popover, paginazioni, breadcrumb, progress bar e molto altro

Javascript – Un set di plugin jQuery utili per animare i componenti Bootstrap - Impareremo ad usarli quando sapremo di più su Javascript

<https://getbootstrap.com/docs/5.3/getting-started/introduction/>



Una predilezione per il mobile



Bootstrap è stato concepito avendo di fronte l'enorme diffusione dei dispositivi mobili (diversi tipi di schermo, diverse risoluzioni)

Per assicurare un rendering appropriato su qualsiasi dispositivo, va aggiunto all'inizio della pagina Web il **viewport** meta-tag

Per disabilitare lo zoom sui dispositivi mobili va aggiunto l'attributo **user-scalable=no**

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,  
user-scalable=no">
```

Page template



HTML5

Fogli di stile

Script Javascript

```
<!DOCTYPE html>
<html>
  <head>
    <title>Biblios :: My online Library</title>
    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0,
      user-scalable=no">
    <!-- Fogli di stile -->
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link href="css/style.css" rel="stylesheet">
    <!-- jQuery e plugin JavaScript -->
    <script src="http://code.jquery.com/jquery.js"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
  </head>
  <body>
```

P
W

HTML responsive



Bootstrap include un potente meccanismo di definizione della griglia di layout che riscalda il posizionamento dei contenuti della pagina su una griglia fino ad un massimo di 12 colonne per ottimizzare la resa di presentazione dei contenuti
(getbootstrap.com/docs/5.3/layout/grid)

Altri elementi (come le immagini e le tabelle) vanno resi *responsive* aggiungendo classi ad-hoc (come **img-responsive** o **table-responsive**)

Per esempio, la classe **img-responsive** imposta automaticamente **width: 100%** e **height: auto**, permettendo di scalare gradualmente le dimensioni delle immagini adattandosi ai dispositivi usati per visualizzare la pagina Web

Grid System



La griglia è costruita a partire da una serie di **div** annidati, dove le classi di stile principali sono:

- ✓ **.container**, che imposta margini a destra e a sinistra della griglia e ne adatta la larghezza alla dimensione del dispositivo utilizzato per visualizzarla
- ✓ **.row**, che rappresenta un'intera riga
- ✓ **.col-xx-N** per ciascuna cella all'interno di una riga, per definire il numero di colonne **N** della griglia su cui si espande la cella (fino ad un massimo di 12) e la tipologia **xx** di dispositivo a cui si applica (tra **xs** = “extra small”, **sm** = “small”, **md** = “medium”, **lg** = “large”)

```
<div class="row">
  <div class="col-xs-12 col-sm-3">
    <div ...8 lines />
  </div>
  <div class="col-xs-12 col-sm-3">
    <div ...6 lines />
  </div>
  <div class="col-xs-12 col-sm-3">
    <div ...6 lines />
  </div>
  <div class="col-xs-12 col-sm-3">
    <div ...6 lines />
  </div>
</div>
```

Bootstrap – Gli altri elementi



Altre classi di stile definite nei file CSS riguardano:

- ✓ Aspetti tipografici (getbootstrap.com/docs/5.3/content/typography/)
- ✓ Tabelle (getbootstrap.com/docs/5.3/content/tables/)
- ✓ Forms (getbootstrap.com/docs/5.3/forms/overview/)
- ✓ Immagini (getbootstrap.com/docs/5.3/content/images/)

... e tanto altro (getbootstrap.com/docs/5.3/getting-started/)

Running example (I)





Home My Library ▾

Home

My online Library

A very simple example of a website created during the Web Programming and Digital Services course. The site lists the books I am currently reading or have read, along with the list of authors who have populated my readings and imagination. The website will continue to grow during this semester, completing itself gradually thanks to the implementation of web technologies that will be introduced in the course. Enjoy!

Books List
Authors List

Home





Home My Library ▾

Home

My online Library

A very simple example of a website created during the Web Programming and Digital Services course. The site lists the books I am currently reading or have read, along with the list of authors who have populated my readings and imagination. The website will continue to grow during this semester, completing itself gradually thanks to the implementation of web technologies that will be introduced in the course. Enjoy!

Sow an act, and you reap a habit; sow a habit, and you reap a character; sow a character, and you reap a destiny.
[Indian proverb]

Home



Running example (II)



The screenshot shows a web application interface. At the top, there is a navigation bar with a small image, 'Home', and 'My Library' (which is highlighted with a grey background). Below the navigation bar, the URL 'Home / Library / Books' is visible. The main title 'My Books List' is centered above a table. To the right of the table, there is a green button labeled 'Create new book' with a pencil icon. The table has two rows. The first row contains the title 'A sapiente vel est.' and the author 'Towne'. To the right of this row are two buttons: a blue 'Edit' button with a pencil icon and a red 'Delete' button with a trash bin icon. The second row contains the title 'Ab preferendis.' and the author 'Schroeder'. To the right of this row are two buttons: a blue 'Edit' button with a pencil icon and a red 'Delete' button with a trash bin icon.

Title	Author		
A sapiente vel est.	Towne	 Edit	 Delete
Ab preferendis.	Schroeder	 Edit	 Delete

Running example (II)



The screenshot shows a web application interface. At the top, there is a navigation bar with a small thumbnail image, the text "Home", and a "My Library" dropdown menu. Below this, the URL "Home / Library / Books" is displayed. The main content area has a title "Insert new book". It contains two input fields: one for "Title" (empty) and one for "Author" containing the value "Bailey". At the bottom are two buttons: a blue "Create" button and a red "Cancel" button.

Home My Library

Home / Library / Books

Home My Library

Home / Library / Books / Add book

Insert new book

Title

Author Bailey

Create

Cancel



Running example (II)



The figure consists of three vertically stacked screenshots of a web application interface. All three screenshots feature a header with a small thumbnail image on the left, followed by 'Home' and 'My Library' with a dropdown arrow. The first two screenshots have a standard grey header, while the third one has a white header.

The first screenshot shows a breadcrumb navigation bar below the header: [Home](#) / [Library](#) / [Books](#).

The second screenshot shows a more detailed breadcrumb: [Home](#) / [Library](#) / [Books](#) / [Add book](#).

The third screenshot shows a confirmation dialog box with the title "Delete book from the list". Below the title is the text "Deleting book. Confirm?". It contains two buttons: "Revert" on the left and "Confirm" on the right. The "Revert" button is associated with the message "The book **will not be removed** from the data base" and a "Cancel" button. The "Confirm" button is associated with the message "The book **will be permanently removed** from the data base" and a red "Delete" button.

Controllare il menu con navbar



<https://getbootstrap.com/docs/5.3/components/navbar/>

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Biblios</a>
    <button class="navbar-toggler" type="button" data-bs-target="#navbarSupportedContent" data-bs-toggle="collapse"
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link" aria-current="page" href="#">../index.html>Home</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle active" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="true">
            My Library
          </a>
          <ul class="dropdown-menu">
            <li><a class="dropdown-item" href="#">../books/books.html>Books List</a></li>
            <li><a class="dropdown-item" href="#">../authors/authors.html>Authors List</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```



Controllare il menu con navbar



<https://getbootstrap.com/docs/5.3/components/navbar/>

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Biblios</a>
    <button class="navbar-toggler" type="button" data-bs-target="#navbarSupportedContent" data-bs-toggle="collapse"
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link" aria-current="page" href="#">index.html>Home</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle active" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="true">
            My Library
          </a>
          <ul class="dropdown-menu">
            <li><a class="dropdown-item" href="#">books/books.html>Books List</a></li>
            <li><a class="dropdown-item" href="#">authors/authors.html>Authors List</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```



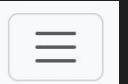
Home My Library ▾

Controllare il menu con navbar



<https://getbootstrap.com/docs/5.3/components/navbar/>

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Biblios</a>
    <button class="navbar-toggler" type="button" data-bs-target="#navbarSupportedContent" data-bs-toggle="collapse"
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link" aria-current="page" href="#">index.html>Home</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle active" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="true">
            My Library
          </a>
          <ul class="dropdown-menu">
            <li><a class="dropdown-item" href="#">books/books.html>Books List</a></li>
            <li><a class="dropdown-item" href="#">authors/authors.html>Authors List</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```



Inserire un breadcrumb



<https://getbootstrap.com/docs/5.3/components/breadcrumb/>

```
<div class="container-fluid d-flex justify-content-end">
  <nav aria-label="breadcrumb">
    <ol class="breadcrumb">
      <li class="breadcrumb-item" aria-current="page"><a href="../index.html">Home</a></li>
      <li class="breadcrumb-item active" aria-current="page">Library</li>
      <li class="breadcrumb-item active" aria-current="page">Books</li>
    </ol>
  </nav>
</div>
```



Inserire icone standardizzate



<https://icons.getbootstrap.com>

```
<!-- Bootstrap Icons -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css">
</head>
```

Icons

Start typing to filter...



```
<a class="btn btn-success"
  href="insertBook.html">
  <i class="bi bi-database-add"></i>
  Create new book</a>
```

 Create new book



Assegnare uno stile alle form



<https://getbootstrap.com/docs/5.3/forms/overview/>

```
<div class="container-fluid">
  <div class="row">
    <div class="col-md-12">
      <form name="book" method="get" action="books.html">
        <div class="form-group row mb-3">
          <div class="col-md-2">
            <label for="title">Title</label>
          </div>
          <div class="col-md-10">
            <input class="form-control" type="text" name="title"/>
          </div>
        </div>
        <div class="form-group row mb-3">
          <div class="col-md-2">
            <label for="author_id">Author</label>
          </div>
```



Assegnare uno stile alle form



<https://getbootstrap.com/docs/5.3/forms/overview/>

```
<div class="container-fluid">
  <div class="row">
    <div class="col-md-12">
      <form name="book" method="get" action="books.html">
        <div class="form-group row mb-3">
          <div class="col-md-2">
            <label for="title">Title</label>
          </div>
          <div class="col-md-10">
            <input class="form-control" type="text" name="title"/>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
```

Insert new book

Title

Author

Create

Cancel



Utilizzare le card



<https://getbootstrap.com/docs/5.3/components/card/>

```
<div class="container-fluid text-center">
  <div class="row">
    <div class="col-md-6 order-md-2">
      <div class="card border-secondary">
        <div class="card-header">
          | Confirm
        </div>
        <div class="card-body">
          <p>
            | The book <strong>will be permanently removed</strong> from the data base
          </p>
          <a class="btn btn-danger" href="#">books.html><i class="bi bi-trash"></i> Delete</a>
        </div>
      </div>
    </div>
  </div>
```



Utilizzare le card



<https://getbootstrap.com/docs/5.3/components/card/>

```
<div class="container-fluid text-center">
  <div class="row">
    <div class="col-md-6 order-md-2">
      <div class="card border-secondary">
        <div class="card-header">
          | Confirm
        </div>
        <div class="card-body">
          <p>
            | The book <strong>will be permanently removed</strong> from the data base
          </p>
          <a class="btn btn-danger" href="#">books.html><i class="bi bi-trash"></i> Delete</a>
        </div>
      </div>
    </div>
  </div>
```

Confirm

The book **will be permanently removed** from the data base

 Delete



Alcuni link utili ...



- ✓ Elementi di Bootstrap 5.3
(getbootstrap.com/docs/5.3/getting-started/)
- ✓ Esempi di Bootstrap 5.3
(getbootstrap.com/docs/5.3/examples)
- ✓ Un set di icone pronte all'uso (icons.getbootstrap.com)
- ✓ Una Community molto attiva (blog.getbootstrap.com)



PROGRAMMAZIONE WEB

IL FRAMEWORK Bootstrap





PROGRAMMAZIONE WEB

EXTENSIBLE MARKUP LANGUAGE (XML)

Prof. Ada Bagozi

ada.bagozi@unibs.it

Ing. Paola Magrino

paola.magrino@unibs.it



Introduzione a XML



eXtensible Markup Language

Formato di file proposto dal W3C per distribuire documenti elettronici sul World Wide Web

Evoluzione:

- ✓ **1986**: Standard Generalized Markup Language (SGML)
ISO 8879-1986
- ✓ **Agosto 1997**: XML Working Draft
- ✓ **Dicembre 1997**: XML 1.0 Proposed Recommendation
- ✓ **Febbraio 1998**: W3C Recommendation

HTML vs XML (I)



- ✓ HTML: insieme fisso di tag
- ✓ XML: standard per creare linguaggi di markup con tag personalizzati (erede di SGML); possono essere usati in qualunque dominio applicativo

```
<h1> Il cosmo di Einstein </h1>
<ul>
  <li> Autore: Kaku Michio
  <li> Editore: Codice
  <li> Anno: 2005
  <li> ISBN: 8875780153
</ul>
```

```
<book>
  <title>Il cosmo di
  Einstein</title>
  <author>Kaku Michio</author>
  <editor>Codice</editor>
  <year>2005</year>
  <isbn>8875780153</isbn>
</book>
```

HTML vs XML (II)



XML e HTML non sono in alternativa, sono nati con scopi diversi!

L'HTML è un caso particolare di XML

- ✓ XML progettato per descrivere i dati (cosa sono i dati)
 - ✓ scambiare e condividere dati e informazioni tra sistemi eterogenei (B2B, B2C,etc.)
 - ✓ creare nuovi linguaggi (WML, MathML...)
- ✓ HTML progettato per visualizzare i dati (come appaiono i dati)
 - ✓ separare i dati dal modo con cui vengono presentati

Recentemente è stato introdotto l'XHTML, che combina l'HTML con le rigide regole sintattiche dell'XML, permettendone una maggiore interoperabilità e l'utilizzo anche su dispositivi a minore capacità

Sintassi XML



- ✓ Ogni documento deve iniziare con `<?xml version="1.0"?>`
- ✓ E' un linguaggio di markup, quindi ne rispetta tutte le regole generali
- ✓ I tag sono "case sensitive"
- ✓ Un documento XML deve avere un tag radice

- ✓ Un documento XML è *ben formato* se soddisfa le regole di sintassi dell'XML

Esempio di documento XML



```
<?xml version="1.0"?>
<elenco>
  <prodotto codice="123">
    <descrizione> Forno </descrizione>
    <prezzo> 1040000 </prezzo>
  </prodotto>
  <prodotto codice="432">
    <descrizione> Frigo </descrizione>
  </prodotto>
</elenco>
```

Dichiarazione iniziale

Elemento radice

Attributo

Elemento con solo testo

Elemento con altri tag annidati

Elementi: modello di contenuto



- ✓ Gli elementi possono avere diversi tipi di contenuto
 - ✓ possono contenere altri elementi annidati (*element content*)

```
<book>
  <publishing year="2005"/>
  <title> Il cosmo di Einstein </title>
  <author> Kaku Michio </author>
  <part> Traduzione a cura di P. Bonini
    <chapter> L'eredità di Albert Einstein </chapter>
  </part>
</book>
```

Elementi: modello di contenuto



- ✓ Gli elementi possono avere diversi tipi di contenuto
 - ✓ possono contenere altri elementi annidati (*element content*)
 - ✓ possono contenere solo testo (*text content*)

```
<book>
  <publishing year="2005"/>
    <title> Il cosmo di Einstein </title>
    <author> Kaku Michio </author>
    <part> Traduzione a cura di P. Bonini
      <chapter> L'eredità di Albert Einstein </chapter>
    </part>
  </book>
```

Elementi: modello di contenuto



- ✓ Gli elementi possono avere diversi tipi di contenuto
 - ✓ possono contenere altri elementi annidati (*element content*)
 - ✓ possono contenere solo testo (*text content*)
 - ✓ possono contenere sia elementi annidati che testo (*mixed content*)

```
<book>
  <publishing year="2005"/>
  <title> Il cosmo di Einstein </title>
  <author> Kaku Michio </author>
  <part> Traduzione a cura di P. Bonini
    <chapter> L'eredità di Albert Einstein </chapter>
  </part>
</book>
```

Elementi: modello di contenuto



- ✓ Gli elementi possono avere diversi tipi di contenuto
 - ✓ possono contenere altri elementi annidati (*element content*)
 - ✓ possono contenere solo testo (*text content*)
 - ✓ possono contenere sia elementi annidati che testo (*mixed content*)
 - ✓ possono essere vuoti (*empty content*) con o senza attributi

```
<book>
    <publishing year="2005"/>
    <title> Il cosmo di Einstein </title>
    <author> Kaku Michio </author>
    <part> Traduzione a cura di P. Bonini
        <chapter> L'eredità di Albert Einstein </chapter>
    </part>
</book>
```

Attributi



```
<prodotto codice="123">  
    <descrizione> Forno </descrizione>  
    <prezzo> 1040000 </prezzo>  
</prodotto>
```

- ✓ Gli elementi possono avere degli attributi
- ✓ I valori vanno racchiusi tra “ ”
- ✓ Differiscono dagli elementi perchè non possono contenere elementi figli
- ✓ Limitazioni nell’uso di attributi
 - ✓ non possono contenere valori multipli
 - ✓ non possono descrivere strutture

Document Type Definition (DTD)



- ✓ Detta il formato comune per una classe di documenti XML, cioè:
 - ✓ gli elementi ammessi
 - ✓ le regole di annidamento degli elementi, gli attributi e il contenuto ammesso per ciascun elemento
- ✓ Scopi:
 - ✓ accordarsi su formato/struttura dei documenti
 - ✓ validare documenti XML secondo certe regole

Un documento XML è **valido** rispetto ad un DTD se rispetta il formato specificato

Tipi di dichiarazioni in un DTD



ELEMENT: introduce il nome dell'elemento e il suo contenuto ammissibile

ATTLIST: specifica gli attributi ammessi per un dato elemento e le proprietà di questi attributi (tipo e vincoli sugli attributi)

ENTITY: simile ad una dichiarazione di costante, si riferisce ad una particolare porzione di documento XML

Dichiarazione di elementi (I)



Elementi contenenti altri elementi figli

```
<!ELEMENT PRODOTTO (DESCRIZIONE)>  
<prodotto><descrizione>...</descrizione></prodotto>
```

Elementi con PCDATA (*parsed character data* = porzione testo qualsiasi)

```
<!ELEMENT DESCRIZIONE (#PCDATA)>  
<descrizione> testo </descrizione>
```

Elementi vuoti

```
<!ELEMENT ARTICOLO EMPTY>  
<articolo/>
```



Dichiarazione di elementi (II)



Contenuto misto

```
<!ELEMENT ARTICOLO (#PCDATA | PRODOTTO)>
<articolo> testo </articolo>
<articolo><prodotto>..</prodotto><articolo>
```

Qualsiasi contenuto

```
<!ELEMENT PARTE ANY>
<parte><sottoparte></sottoparte><parte>
<parte><prodotto></prodotto></parte>
```

Occorrenze di un elemento



1 volta

```
<!ELEMENT PRODOTTO (DESCRIZIONE)>
```

1 o più volte

```
<!ELEMENT LISTA (PRODOTTO+)>
```

0 o più volte

```
<!ELEMENT LISTA (PRODOTTO*)>
```

0 o 1 volta

```
<!ELEMENT PRODOTTO (DESCRIZIONE?)>
```

Dichiarazione di attributi



Per ogni elemento il DTD dice:

- ✓ quali attributi può avere il tag
- ✓ che valori può assumere ciascun attributo
- ✓ eventuali vincoli sulla cardinalità degli attributi
- ✓ qual è il valore di default

Esempio di dichiarazione di attributo:

```
<!ATTLIST PRODOTTO  
    codice ID #REQUIRED  
    label CDATA #IMPLIED  
    status (disponibile | terminato)  
    "disponibile">
```

Tipi di attributi



CDATA: stringa

ID: chiave unica

IDREF, IDREFS: riferimento ad uno o più ID nel documento

ENTITY, ENTITIES: nome di una o più entità

NMTOKEN, NMTOKENS: caso ristretto di CDATA (una stringa di una o più parole separate da spazi)

```
codice ID #REQUIRED
      label CDATA #IMPLIED
      status (disponibile|terminato) 'disponibile'
```

Vincoli sugli attributi



#REQUIRED: il valore deve essere specificato

#IMPLIED: l'attributo può avere un valore e il valore di default non è definito

“val”: il valore dell'attributo è “val” se nient'altro è specificato

#FIXED “val”: il valore, se presente, deve coincidere con “val”

```
codice ID      #REQUIRED
label CDATA   #IMPLIED
status (disponibile|terminato) "disponibile"
```



Dichiarazione di un'entità



Analoghe alle dichiarazioni di macro con **#define** in C

```
<!ENTITY ATI "ArborText, Inc.">
<!ENTITY boilerplate SYSTEM
"/standard/legalnotice.xml">
```

Le entità possono essere

- ✓ interne (**&ATI;**)
- ✓ esterne (**&boilerplate;**)

Documenti XML con DTD



```
<?XML version="1.0" standalone="no"?>  
  
<!DOCTYPE elenco SYSTEM "libro.dtd">  
<!DOCTYPE elenco [  
    <!ELEMENT ELENCO (PRODOTTO+)>  
    <!ELEMENT PRODOTTO (DESCRIZIONE, PREZZO?)>  
    <!ELEMENT DESCRIZIONE (#PCDATA)>  
    <!ELEMENT PREZZO (#PCDATA)>  
    <!ATTLIST PRODOTTO codice ID #REQUIRED>  
>  
  
<elenco>...</elenco>
```

DTD esterno

Esempio di DTD



```
<!ELEMENT ELENCO (PRODOTTO+)>
<!ELEMENT PRODOTTO (DESCRIZIONE, PREZZO?)>
<!ELEMENT DESCRIZIONE (#PCDATA)>
<!ELEMENT PREZZO (#PCDATA)>
<!ATTLIST PRODOTTO codice ID #REQUIRED>
```

```
<elenco>
    <prodotto codice="123">
        <descrizione> Forno </descrizione>
        <prezzo> 1040000 </prezzo>
    </prodotto>
    <prodotto codice="432">
        <descrizione> Frigo </descrizione>
    </prodotto>
</elenco>
```

Esempio di DTD



```
<!ELEMENT ELENCO (PRODOTTO+)>
<!ELEMENT PRODOTTO (DESCRIZIONE, PREZZO?)>
<!ELEMENT DESCRIZIONE (#PCDATA)>
<!ELEMENT PREZZO (#PCDATA)>
<!ATTLIST PRODOTTO codice ID #REQUIRED>
```

<elenco>

NOTA: un DTD NON è un documento XML

```
<descrizione> Forno </descrizione>
<prezzo> 1040000 </prezzo>
</prodotto>
<prodotto codice="432">
    <descrizione> Frigo </descrizione>
    </prodotto>
</elenco>
```

XML Schema



Storia: inizialmente proposto da Microsoft, è divenuto W3C recommendation (maggio 2001)

Scopo: definire gli elementi e la composizione di un documento XML

Un XML Schema definisce regole riguardanti:

- ✓ Elementi
- ✓ Attributi
- ✓ Gerarchia degli elementi
- ✓ Sequenza di elementi figli
- ✓ Cardinalità di elementi figli
- ✓ Tipi di dati per elementi e attributi
- ✓ Valori di default per elementi e attributi

Elementi semplici



Possono contenere solo testo (nessun sottoelemento o attributo)

Definizione di elementi semplici:

```
<xs:element name="nome" type="tipo"/>  
<xs:element name="nome" type="tipo" default="xyz"/>  
<xs:element name="nome" type="tipo" fixed="xyz" />
```

Esempi di definizione di elementi semplici in XSD

```
<xs:element name="età" type="xs:integer"/>  
<xs:element name="cognome" type="xs:string"/>
```

Elementi semplici in un documento XML

```
<età> 65 </età>  
<cognome> Rossi </cognome>
```

Valore di
default o fisso

Elementi complessi



```
<xs:element name="book">
  <xs:complexType>
    . . . element content . .
  </xs:complexType>
</xs:element>
```

Vincoli di cardinalità



Numero di occorrenze:

maxOccurs: max numero di occorrenze

minOccurs: min numero di occorrenze

Se non specificati: 1 e 1 sola occorrenza

```
<xs:element name="Nome" type="xs:string" maxOccurs="4" minOccurs="1"/>
```

Raggruppamento



Per definire gruppi di elementi (o attributi), tra loro correlati (group name)

```
<xs:group name="completeName">
  <xs:sequence>
    <xs:element name="Cognome" type="xs:string"/>
    <xs:element name="Nome" type="xs:string"/>
  </xs:sequence>
<xs:group>
...
<xs:group ref="completeName"/>
```

Restrizioni



E' possibile imporre delle restrizioni sui valori dei tipi semplici tramite l'uso di "*facet*"

- ✓ Valore max/min
- ✓ Enumerazione di valori
- ✓ Pattern di caratteri ammessi
- ✓ Lunghezza di liste di valori

Le restrizioni sono applicate al range di valori ammessi per questo tipo

```
<xs:attribute name="format">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="paperback"/>
      <xs:enumeration value="hardback"/>
    <xs:restriction>
      <xs:simpleType>
    </xs:attribute>
```

Facet + valore

Riferimenti ad altri elementi



Attraverso l'attributo “**ref**” ci si può riferire ad un elemento definito altrove

```
...
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="to"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
...
<xs:element name="to"
  type="xs:string"/>
```

Esempio di XML Schema



```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="head" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

XHTML



Si tratta di una versione più “controllata” dell'HTML, che segue le regole dell'XML

- ✓ annidamento corretto degli elementi `<i>...</i>`
- ✓ tag di chiusura `<p>...</p>`
- ✓ gestione ordinata dei tag vuoti `
`
- ✓ *case sensitivity* per i nomi dei tag e degli attributi (minuscolo)
- ✓ Valori di attributi tra virgolette ed esplicativi
`checked="checked"`

Compatibilità cross-browser (I)



I Browser supportano due modalità di rendering: **Standards mode** and **Quirks mode**

- ✓ Lo **Standards mode** lavora seguendo il più possibile le specifiche del W3C, quindi in maniera (quasi) indipendente dal browser
- ✓ Il **Quirks mode** segue le regole di formattazione dello specifico browser, con le sue limitazioni ed estensioni.

La modalità Quirks esiste per rendere i browser compatibili con i vecchi siti web, che erano sviluppati con codice molto browser-dipendente. Oggi, è necessario sviluppare i nuovi siti in modalità Standards

Compatibilità cross-browser (II)



(!) Di default i browser usano la modalità Quirks

Per entrare in modalità Standards occorre inserire all'inizio del documento una dichiarazione doctype come quella che segue

✓ Per usare l'XHTML transitional:

```
<? HTML5 introduce una semplificazione  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
Tr<!DOCTYPE html>  
transitional.dtd">
```

✓ Per usare l'XHTML strict:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Per saperne di più...



✓ Tutorial W3C

- Su XML: <https://www.w3schools.com/xml>
- Su DTD: <https://www.w3schools.com/xml>
- Su XSD: https://www.w3schools.com/xml/schema_intro.asp
- Su XHTML: <https://www.w3.org/xhtml1/>



PROGRAMMAZIONE WEB

EXTENSIBLE MARKUP LANGUAGE (XML)

Prof. Ada Bagozi

ada.bagozi@unibs.it

Ing. Paola Magrino

paola.magrino@unibs.it





PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - PHP

Prof. Ada Bagozi

ada.bagozi@unibs.it



Cos'è il PHP?

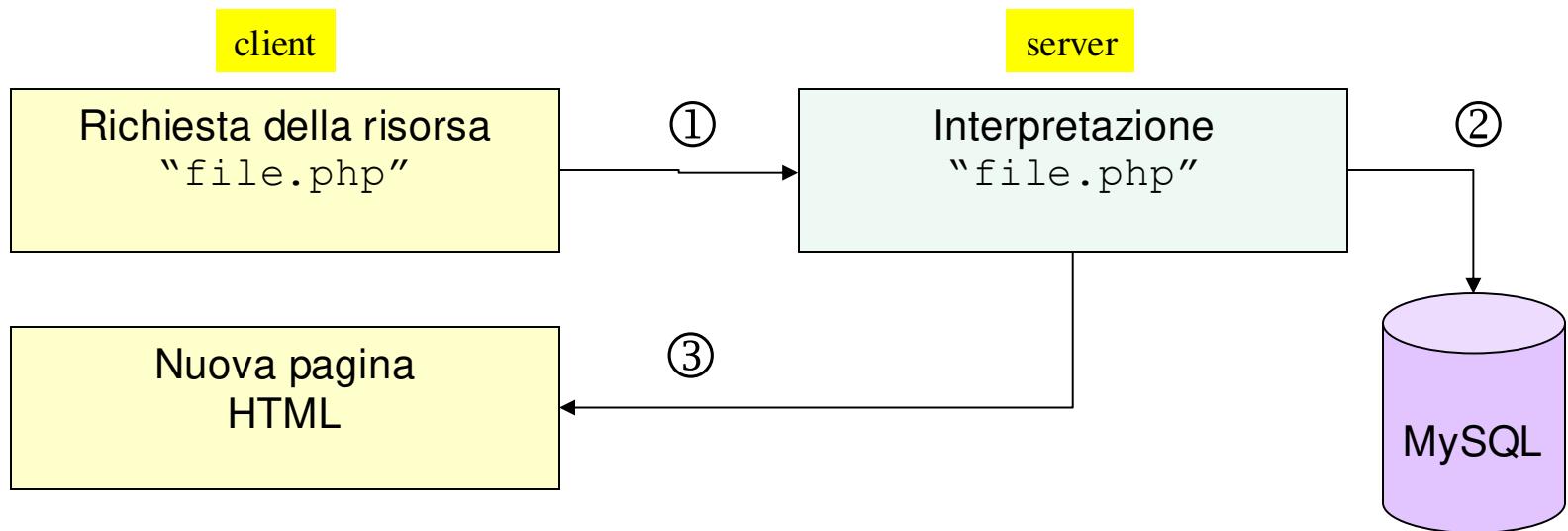


- ✓ *PHP: Hypertext Preprocessor* è un linguaggio di programmazione utilizzato prevalentemente per la programmazione di pagine Web
- ✓ Alcune caratteristiche “tecniche”
 - ✓ è un linguaggio di *scripting*
 - ✓ è *interpretato* (non *compilato*)
 - ✓ è *server-side*
 - ✓ è *debolmente tipizzato*
 - ✓ consente la *programmazione Object Oriented*
- ✓ Ed inoltre.. è *Open Source*

Esecuzione di codice PHP



- ✓ Il codice PHP viene inserito in una pagina HTML (sul server)
- ✓ Il codice PHP viene interpretato (sul server)
- ✓ Viene generata una nuova pagina HTML



Struttura del codice PHP (I)



- ✓ Il codice PHP è racchiuso tra due tag `<?php` e `?>`

```
<?php  
    ... codice PHP ...  
?>
```

- ✓ Le istruzioni PHP terminano con il punto e virgola

```
echo 'Hello ';  
echo 'world!';
```

- ✓ I commenti hanno tre diversi stili

```
/* Autore: Devis Bianchini  
Ultima modifica: 15 aprile 2011
```

```
*/
```

```
echo '<p>Ordine gestito.</p>'; // Inizio stampa ordine  
echo '<p>Ordine gestito.</p>'; # Inizio stampa ordine
```

Variabili



- ✓ Nome di una variabile (identificatore):
 - ✓ Inizia con il simbolo **\$**
 - ✓ Formato da lettere, cifre e underscore ‘_’
 - ✓ Lunghezza illimitata
 - ✓ Non può iniziare con una cifra
 - ✓ Case sensitive: **\$quantity** ≠ **\$Quantity**
- ✓ Una variabile viene creata nel momento in cui viene assegnata la prima volta

```
$quantity = 0;  
$costo = 0.00;  
...  
$quantity = $q;
```

Tipi di variabili



Tipo	Natura del dato
<code>integer</code>	Numeri interi
<code>double</code>	Numeri reali
<code>string</code>	Stringhe di caratteri
<code>boolean</code>	Valori logici (<code>true</code> o <code>false</code>)
<code>array</code>	Vettori di dati

Esistono altri due tipi “speciali”:

- ✓ **NULL**: variabili cui non è assegnato un valore o sono state assegnate con NULL
- ✓ **Resource**: rappresentano risorse esterne (esempio: connessione al database)

Tipizzazione delle variabili in PHP



- ✓ Le variabili sono molto “elastiche” nell’assegnamento del tipo di dati

```
$quantity = 0;  
$quantity = 'Hello' ;
```

- ✓ Casting

```
$quantity = 0;  
$costo = (double) $quantity;
```

Tipizzazione debole e array (I)



```
<?php
$sv1 = array();

$sv2 = array("lunedì","martedì","mercoledì");
echo $sv2[2] // stampa mercoledì', ossia l'elemento in posizione 2

$sv3 = array(
    0=>"lunedì",
    1=>"martedì",
    2=>"mercoledì");
// $v3 ha lo stesso contenuto di $v2

$sv4 = array(
    "mela"=>"frutta",
    "pera"=>"frutta",
    "carciofo"=>"verdura");
// $v4 è un array associativo
echo $sv4["pera"] // stampa frutta
?>
```

Tipizzazione debole e array (II)



```
<?php
    $v5 = array(
        3=>"giovedi",
        4=>"venerdi",
        5=>"sabato");
        // array in cui il primo indice è 3
    $v5[4] = "venerdi"; // modifica un valore
    $v5[] = "domenica"; // aggiunge un elemento in coda
    $v5[1] = "martedì"; // aggiunge un elemento in posizione 1

    // Un vettore può avere altri vettori come elementi
    // Inoltre il tipo degli elementi può essere eterogeneo
    $automobili = array(
        "Paperone"=>array(1,213),
        "Paperino"=>313,
        "Qui"=>"nessuna");

    echo count($automobili); // stampa la lunghezza dell'array, cioè 3
?>
```

Variabili predefinite



PHP mette a disposizione un gran numero di variabili predefinite

Sono principalmente dedicate a descrivere il server su cui è in funzione, le richieste HTTP, i dati inseriti nei campi di una form

Alcune variabili predefinite possono essere dipendenti dalla piattaforma

```
<?php  
echo "<a href=\"http://",  
      $_SERVER["HTTP_HOST"],      // nome del sito  
      $_SERVER["PHP_SELF"],       // nome dello script  
      "\">Link a me stesso</a>\n";  
?>
```

Lavorare con le variabili



È possibile inserire direttamente una variabile semplice in una stringa

```
echo "$qbanane banane <br />"; ➔ 2 banane
```

Tuttavia, le stringhe tra apici semplici non applicano alcuna sostituzione:

```
echo '$qbanane banane <br />'; ➔ $qbanane banane
```

Costanti



Mantengono un valore (come una variabile), che però non può essere cambiato

```
define(<nome-costante>, <valore>)
```

```
define(PERCENTUALE_SCONTI, 25);
```

Operatori (I)



Operatori aritmetici

Simbolo	Nome	Esempio
+	Addizione	$\$a + \b
-	Sottrazione	$\$a - \b
*	Moltiplicazione	$\$a * \b
/	Divisione	$\$a / \b
%	Modulo	$\$a \% \b

Operatore di concatenazione delle stringhe

```
$a = 'Strumenti Musicali ';  
$b = 'in Franciacorta';  
$titolo = $a . $b;
```

Operatori (II)



Operatore di **assegnamento**

<variabile> = <espressione>

`$b = 6 + ($a = 5);`

Operatori di **assegnamento combinato**

Simbolo	Esempio	Equivalente a
<code>+=</code>	<code>\$a += \$b</code>	<code>\$a = \$a + \$b</code>
<code>--=</code>	<code>\$a --= \$b</code>	<code>\$a = \$a - \$b</code>
<code>*=</code>	<code>\$a *= \$b</code>	<code>\$a = \$a * \$b</code>
<code>/=</code>	<code>\$a /= \$b</code>	<code>\$a = \$a / \$b</code>
<code>%=</code>	<code>\$a %= \$b</code>	<code>\$a = \$a % \$b</code>
<code>.=</code>	<code>\$a .= \$b</code>	<code>\$a = \$a . \$b</code>

Operatori (III)



Operatore di **incremento** (++) e **decremento** (--)

Sono entrambi disponibili in due forme

- ✓ Prima della variabile `$a = 5;
echo ++$a;`

prima **\$a** viene incrementato (6) e poi visualizzato (6)

- ✓ Dopo la variabile `$a = 5;
echo $a++;`

prima **\$a** viene visualizzato (5) e poi incrementato (6)

```
$a = 3;  
$b = 6;  
echo ($a--) * (++$b);
```



```
$a: 2  
$b: 7  
echo: 21
```

Operatori (IV)



Operatore di confronto

Simbolo	Nome	Esempio
<code>==</code>	Uguaglianza	<code>\$a == \$b</code>
<code>===</code>	Identità	<code>\$a === \$b</code>
<code>!=</code>	Disuguaglianza	<code>\$a != \$b</code>
<code>!==</code>	Non identità	<code>\$a !== \$b</code>
<code><</code>	Minore di	<code>\$a < \$b</code>
<code>></code>	Maggiore di	<code>\$a < \$b</code>
<code><=</code>	Minore o uguale	<code>\$a <= \$b</code>
<code>>=</code>	Maggiore o uguale	<code>\$a >= \$b</code>

Operatore identità: operandi uguali e dello stesso tipo



`$a = 0;
$b = '0';`



`$a == $b: true
$a === $b: false`

Operatori (V)



Operatori logici

Simbolo	Nome	Esempio	Risultato															
!	NOT	<code>! \$a</code>	<table border="1"><tr><td><code>\$a</code></td><td><code>! \$a</code></td></tr><tr><td>true</td><td>false</td></tr><tr><td>false</td><td>true</td></tr></table>	<code>\$a</code>	<code>! \$a</code>	true	false	false	true									
<code>\$a</code>	<code>! \$a</code>																	
true	false																	
false	true																	
&&	AND	<code>\$a && \$b</code>	<table border="1"><tr><td><code>\$a</code></td><td><code>\$b</code></td><td><code>\$a && \$b</code></td></tr><tr><td>true</td><td>true</td><td>true</td></tr><tr><td>true</td><td>false</td><td>false</td></tr><tr><td>false</td><td>true</td><td>false</td></tr><tr><td>false</td><td>false</td><td>false</td></tr></table>	<code>\$a</code>	<code>\$b</code>	<code>\$a && \$b</code>	true	true	true	true	false	false	false	true	false	false	false	false
<code>\$a</code>	<code>\$b</code>	<code>\$a && \$b</code>																
true	true	true																
true	false	false																
false	true	false																
false	false	false																
	OR	<code>\$a \$b</code>	<table border="1"><tr><td><code>\$a</code></td><td><code>\$b</code></td><td><code>\$a \$b</code></td></tr><tr><td>true</td><td>true</td><td>true</td></tr><tr><td>true</td><td>false</td><td>true</td></tr><tr><td>false</td><td>true</td><td>true</td></tr><tr><td>false</td><td>false</td><td>false</td></tr></table>	<code>\$a</code>	<code>\$b</code>	<code>\$a \$b</code>	true	true	true	true	false	true	false	true	true	false	false	false
<code>\$a</code>	<code>\$b</code>	<code>\$a \$b</code>																
true	true	true																
true	false	true																
false	true	true																
false	false	false																

Operatori (VI)



Operatore condizionale (ternario)

```
( <condizione> ? <valore se vera> : <valore se falsa> )
```

```
echo ($voto >= 18 ? 'promosso' : 'bocciato')
```

```
$max = ($a >= $b ? $a : $b)
```

```
$max3 = ($a >= $b && $a >= $c ?  
          $a : ($b >= $c ? $b : $c))
```

```
echo 'Hai ordinato '. $qbanane .' banan'  
      . ($qbanane == 1 ? 'a' : 'e')
```

Type juggling



Alcune conversioni di tipo avvengono automaticamente in base agli operatori utilizzati nelle espressioni

La conversione non modifica il tipo degli operandi, che rimangono inalterati

```
<?php  
$a = "0";  
  
$a .= 2;  
  
$a += 2;  
  
$b = $a + 1.3;  
?>
```

Funzioni di variabili



Per testare o modificare lo stato di una variabile si possono usare le seguenti funzioni:

- ✓ **isset(\$var)**: **true** se **\$var** esiste, altrimenti **false**
- ✓ **isset(\$a, \$b, \$c, ...)**: **true** se tutte esistono, altrimenti **false**
- ✓ **unset(\$var)**: elimina **\$var**
- ✓ **empty(\$var)**: **true** se **\$var** non esiste o ha valore zero, altrimenti **false**

```
$qbanane = 3;  
echo 'echo1: ' . isset($qbanane) . '<br />' ; // TRUE  
echo 'echo2: ' . isset($qmele) . '<br />' ; // FALSE  
echo 'echo3: ' . empty($qbanane) . '<br />' ; // FALSE  
echo 'echo4: ' . empty($qmele) . '<br />' ; // TRUE
```

Funzioni di accesso al tipo



- ✓ **`is_boolean($a)`**: verifica se la variabile contiene un valore booleano
- ✓ **`is_integer($a)`**: verifica se la variabile contiene un numero intero
- ✓ **`is_float($a)`, `is_double($a)`, `is_array($a)`, `is_resource($a)`**
- ✓ **`is_null($a)`**: verifica se la variabile contiene il valore `null`
- ✓ **`is_numeric($a)`**: verifica se il contenuto della variabile è compatibile con un valore numerico (ossia se è un numero o una stringa convertibile in numero)
- ✓ **`gettype($a)`**: restituisce il nome del tipo della variabile sotto forma di stringa

L'istruzione if



```
if(<condizione>)
    <istruzioni>
```

Il blocco <istruzioni> viene eseguito solo se <condizione> è vera

```
if($quantita_totale == 0)
    echo 'Non hai ordinato alcun articolo! <br />';

if($costo_totale >= 30000)
{
    echo 'Hai diritto ad uno sconto del 10% <br />';
}
```

L'istruzione else



```
else  
<istruzioni>
```

Il blocco <istruzioni> viene eseguito solo se la condizione del precedente **if** è falsa

```
if($quantita_totale == 0)  
    echo "Non hai ordinato alcun articolo! <br />";  
else  
{  
    echo 'Ecco la lista degli articoli: <br />';  
    echo '<ul>';  
    echo "<li>$qpomodori pomodori</li>";  
    echo "<li>$qbanane banane</li>";  
    echo "<li>$qmele mele</li>";  
    echo "</ul>";  
}
```

L'istruzione elseif



È usato quando si hanno più di due rami decisionali:

```
elseif <condizione>
      <istruzioni>
```

```
if($costo_totale <= 10000)
    $sconto = 0.0;
elseif($costo_totale > 10000 && $costo_totale <= 20000)
    $sconto = 0.10;
elseif($costo_totale > 20000 && $costo_totale <= 40000)
    $sconto = 0.20;
elseif($costo_totale > 40000 && $costo_totale <= 80000)
    $sconto = 0.25;
else
    $sconto = 0.30;
```

Viene eseguito solo il blocco di istruzioni corrispondente alla prima condizione vera

Se nessuna condizione vera, viene eseguito il blocco della **else** (se specificata)

L'istruzione switch



```
switch (<condizione>) {  
    case <valore1>: <codice> break;  
    case <valore2>: <codice> break;  
    ....  
    default: <codice>; break;  
}
```

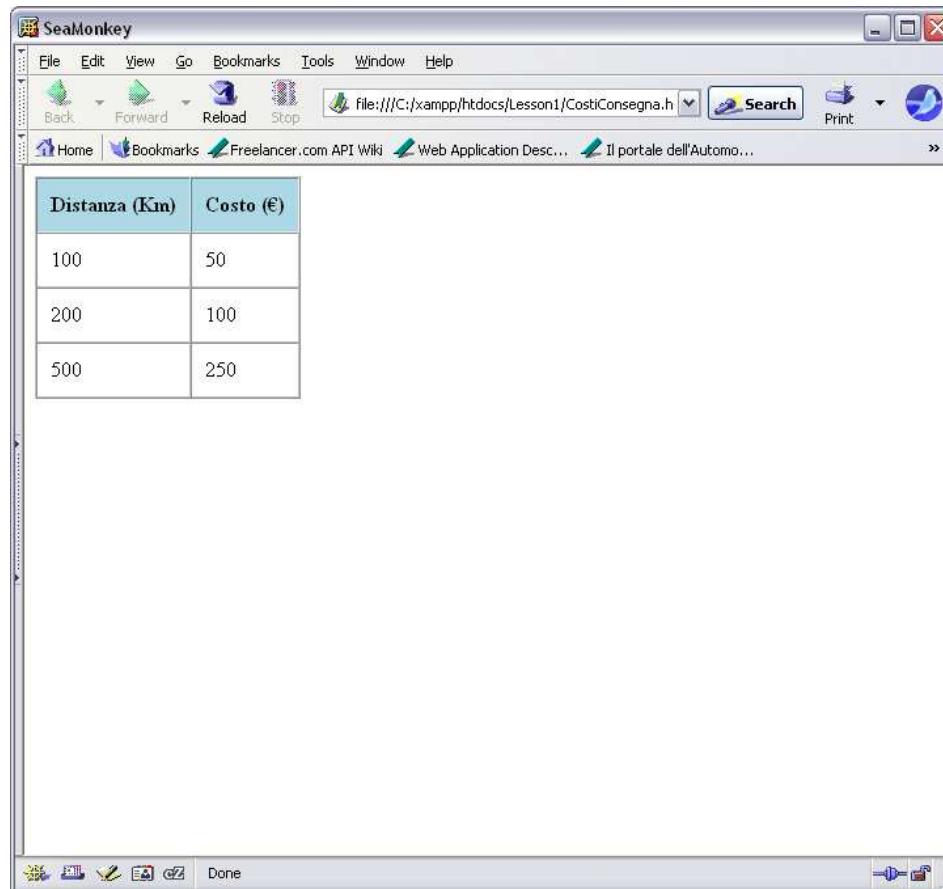
Il costrutto **switch** permette di diversificare l'operato del programma al valore dell'espressione testata

```
switch ($nome)  
{  
    case 'Luca':  
    case 'Giorgio':  
    case 'Franco':  
        echo "Ciao, vecchio amico!"; break;  
    case 'Mario': echo "Ciao, Mario!"; break;  
    default: print "Benvenuto, chiunque tu sia";  
}
```

Iterazioni



Sono utilizzate quando è necessario eseguire più volte lo stesso blocco di istruzioni



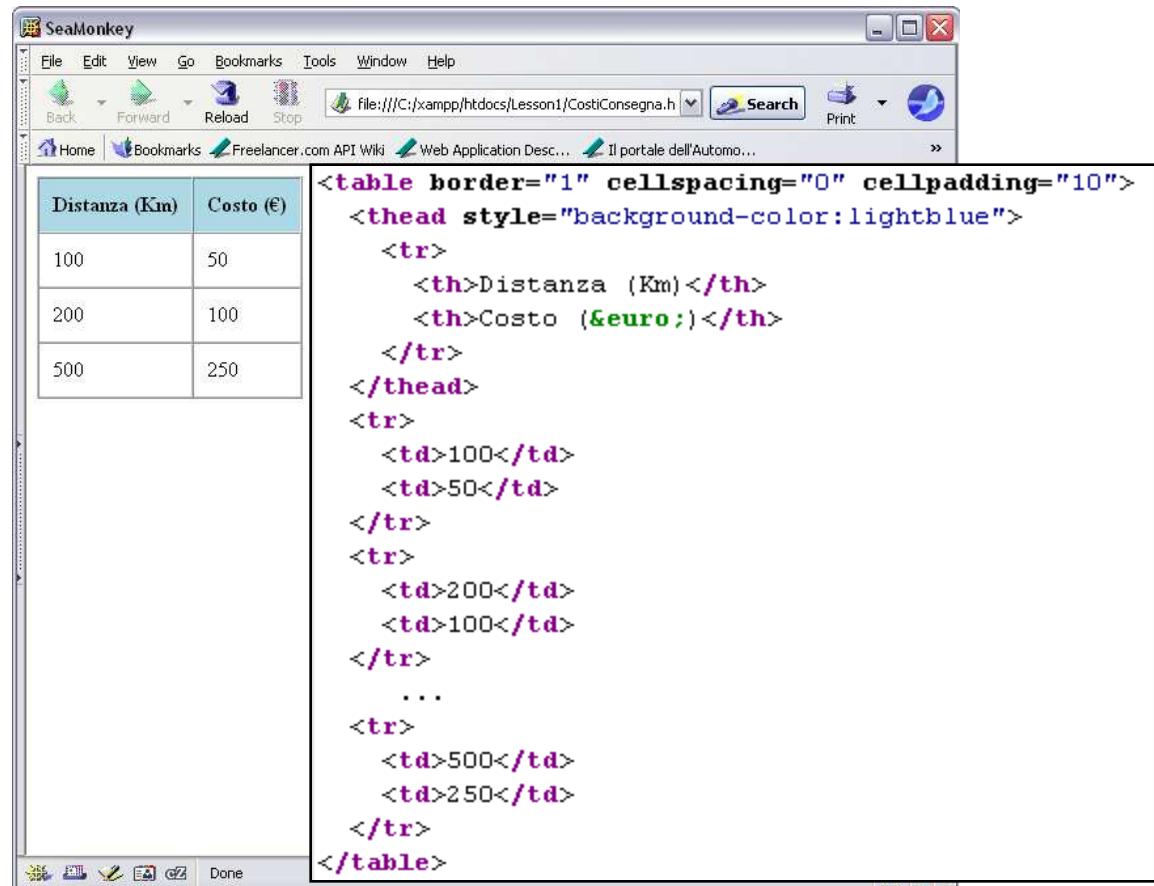
The screenshot shows a SeaMonkey web browser window with the title "SeaMonkey". The address bar displays "file:///C:/xampp/htdocs/Lesson1/CostiConsegna.html". The main content area contains a table with three rows:

Distanza (Km)	Costo (€)
100	50
200	100
500	250

Iterazioni



Sono utilizzate quando è necessario eseguire più volte lo stesso blocco di istruzioni



The screenshot shows a SeaMonkey browser window displaying an HTML table. The table has two columns: 'Distanza (Km)' and 'Costo (€)'. The data rows are 100, 50; 200, 100; and 500, 250. To the right of the table, the source code is shown, demonstrating the use of loops (tr, td) to generate the table rows.

Distanza (Km)	Costo (€)
100	50
200	100
500	250

```
<table border="1" cellspacing="0" cellpadding="10">
<thead style="background-color:lightblue">
<tr>
<th>Distanza (Km)</th>
<th>Costo (&euro;)</th>
</tr>
</thead>
<tr>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
</tr>
...
<tr>
<td>500</td>
<td>250</td>
</tr>
</table>
```

Ciclo while



```
while (<condizione>)
    <istruzioni>
```

Il blocco <istruzioni> viene eseguito fino a quando <condizione> è vera

```
<table border="1" cellspacing="0" cellpadding="10">
    <thead style="background-color:lightblue">
        <tr>
            <th>Distanza (Km)</th>
            <th>Costo (&euro;)</th>
        </tr>
    </thead>
    <?php
        $distanza = 100;
        while($distanza <= 500)
        {
            echo '<tr><td>' . $distanza . '</td>';
            echo '<td>' . ($distanza/2) . '</td></tr>';
            $distanza += 100;
        }
    ?>
</table>
```

Ciclo for



```
for (<istruzione1>; <condizione>; <istruzione2>)
    <istruzioni>
```

Il significato è esprimibile in termini di ciclo **while** come segue

```
<istruzione1>;
while (<condizione>)
{
    <istruzioni>
    <istruzione2>;
}
```

Ciclo for



```
for (<istruzione1>; <condizione>; <istruzione2>)
    <istruzioni>
```

Il significato è esprimibile in termini di ciclo **while** come segue

```
<istruzione1>;
while (<condizione>)
{
    <istruzioni>
    <istruzione2>;
}
```

```
<table border="1" cellspacing="0" cellpadding="10">
    <thead style="background-color:lightblue">
        <tr>
            <th>Distanza (Km)</th>
            <th>Costo (€uro :)</th>
        </tr>
    </thead>
<?php
    for($distanza = 100; $distanza <= 500; $distanza += 100)
    {
        echo '<tr><td>' . $distanza . '</td>';
        echo '<td>' . ($distanza/2) . '</td></tr>';
    }
?>
</table>
```

Ciclo do..while



```
do
    <istruzioni>
    while (<condizione>)
```

È simile al ciclo **while**, ma <condizione> viene testata dopo l'esecuzione di <istruzioni>

```
<table border="1" cellspacing="0" cellpadding="10">
    <thead style="background-color:lightblue">
        <tr>
            <th>Distanza (Km)</th>
            <th>Costo (&euro;)</th>
        </tr>
    </thead>
    <?php
        $distanza = 100;
        do
        {
            echo '<tr><td>' . $distanza . '</td>';
            echo '<td>' . ($distanza/2) . '</td></tr>';
            $distanza += 100;
        }while($distanza <= 500)
    ?>
</table>
```

Ciclo foreach



```
foreach (array as $valore)  
istr
```

Il ciclo si ripete tante volte quanti sono gli elementi dell'array e all'interno del ciclo ogni volta è disponibile, nella variabile **\$valore**, il valore dell'elemento corrispondente all'iterazione

```
foreach (array as $chiave=>$valore)  
istr
```

Questa versione è particolarmente utile per gli array associativi; oltre al valore è presente anche la chiave dell'elemento, presente nella variabile **\$chiave**

Funzioni



```
function nome_funzione ($arg1, $arg2)
{
    // codice della funzione (corpo)

    // l'istruzione return serve per restituire un valore
    // come risultato ed è opzionale
    return <risultato>;
}
```

Funzioni con argomenti di default



```
function nome_funzione ($arg1, $arg2="default")
{
    // codice della funzione (corpo)
return <risultato>;
}

. . .

$ris1 = nome_funzione("primo","secondo");      // senza l'uso del
                                                // default
$ris2 = nome_funzione("primo");                // con l'uso del default
```

Esempio: gestione di un ordine



```
<body>
    <h1>Gestione Ordine<h1>
    <?php
        echo '<p>Ordine gestito alle ore ';
        echo date("H:i, jS F");
        echo '. Grazie per aver scelto i nostri prodotti.</p>';
    ?>
</body>
```

Argomento della funzione `date()` = stringa che specifica il formato:

H = ore (in formato 24-ore)

i = minuti

j = giorno del mese

S = suffisso ordinale (tipicamente, *th*)

F = nome del mese

09:52, 15th April

Visibilità delle variabili



La **visibilità** (o **scope**) di una variabile **\$v** è la porzione del codice in cui è visibile **\$v**:

- ✓ Le variabili **superglobali** sono visibili ovunque nello script (es., **\$_SERVER**)
- ✓ Le costanti sono visibili ovunque nello script
- ✓ Le variabili **globali** sono visibili ovunque nello script, tranne che nelle funzioni
- ✓ Le variabili dichiarate globali nelle funzioni si riferiscono alle omonime variabili dichiarate globali fuori delle funzioni
- ✓ Le variabili dichiarate **statiche** in una funzione sono visibili solo nella funzione ma mantengono il loro valore tra una chiamata e l'altra della funzione
- ✓ Le variabili **locali** create in una funzione sono visibili solo nella funzione e scompaiono al termine della esecuzione della funzione

Visibilità delle variabili: esempio



```
$a = 1;  
function nome_funzione ()  
{  
    global $a;      // senza questa riga la funzione non stampa  
                   // niente  
    echo $a;  
}  
  
.  
.  
.  
nome_funzione();
```

Classi e oggetti (I)



- ✓ Nel linguaggio PHP il concetto di classe è quello tradizionale dei linguaggi di programmazione ad oggetti

```
class nome_classe
{
    var variabili_comuni_alla_classe;

    function nome_classe(parametro1="valore_di_default1",...) {...}

    function nome_metodo1(parametro1, parametro2, ...) {...}

    function nome_metodo2(parametro1, parametro2, ...) {...}

    ...
}
```

Classi e oggetti (II)



- ✓ La gestione delle classi e della programmazione ad oggetti è stata enormemente potenziata nell'ultima versione di PHP (PHP 5)
 - ✓ sono stati introdotti i modificatori di visibilità **private**, **public** e **protected**, su cui valgono le comuni regole della programmazione a oggetti
 - ✓ sono stati introdotti i metodi **construct** (nome univoco per il costruttore) e **destruct** (nome univoco metodo distruttore)
 - ✓ è possibile passare gli oggetti per **riferimento** e non per **valore**

Costruttori e distruttori



- ✓ PHP 5 ha introdotto un nome standardizzato per i **costruttori**, **construct**
 - ✓ se la classe viene rinominata, non è necessario modificare anche il nome del suo costruttore
- ✓ Un'altra novità in PHP 5 è l'introduzione del metodo **distruttore**, **destruct()**
 - ✓ utile per operazioni di pulizia, come l'eliminazione di file temporanei o la chiusura di connessioni a database (**garbage collection**)
 - ✓ la garbage collection avviene nel momento in cui viene eliminato l'ultimo riferimento all'oggetto

Classi e oggetti: esempio (I)



```
<?php
2 references | 0 implementations
class Book
{
    3 references
    private $id;
    3 references
    private $title;
    3 references
    private $author_lastName;
    3 references
    private $authorID;

    2 references | 0 overrides
    public function __construct($identifier, $book_title, $author_name, $author_id)
    {
        $this->id = $identifier;
        $this->title = $book_title;
        $this->author_lastName = $author_name;
        $this->authorID = $author_id;
    }

    2 references | 0 overrides
    function getId() {
        return $this->id;
    }
}
```



Classi e oggetti: esempio (II)



```
<?php
2 references | 0 implementations
class Author
{
    3 references
    private $id;
    3 references
    private $firstName;
    3 references
    private $lastName;

    2 references | 0 overrides
    public function __construct($identifier, $first_name, $last_name)
    {
        $this->id = $identifier;
        $this->firstName = $first_name;
        $this->lastName = $last_name;
    }

    0 references | 0 overrides
    public function getId()
>    ...
}
```



Istanziazione



- ✓ Una volta definita la struttura della classe, è possibile istanziare uno o più oggetti di quel tipo
 - ✓ lasciando i valori di default
 - ✓ specificando nuovi parametri al momento dell'istanziazione

```
$author = new Author(2, "Umberto", "Eco");
$book = new Book(5, "Il nome della rosa", $author->getLastName(), $author->getId());
```

Invocazione dei metodi di una classe



- ✓ Dopo aver istanziato la classe, i metodi della classe possono essere invocati per andare a modificare le variabili di stato della classe stessa
 - ✓ le invocazioni dei metodi su una istanza sono indipendenti dalle invocazioni di metodi su istanze diverse

```
echo $author->getFirstName() . " " . $author->getLastName();  
  
$book->setTitle("My second life");
```

PHP 5 e oggetti



- ✓ In PHP 5 è stato anche potenziato il meccanismo dell'ereditarietà (parola chiave `extends`)
 - ✓ sono state introdotte le *interfacce*
 - ✓ è stato introdotto l'operatore `instanceof`
 - ✓ è possibile definire metodi e classi come `final`
 - ✓ è stato introdotto l'uso di `static` per definire proprietà e metodi statici

Interfacce



Grazie all'uso delle interfacce, viene superato il vincolo dell'ereditarietà da una singola classe; infatti una classe può implementare più interfacce

```
interface interfaccia_1 {
    /* metodi astratti */
    function funzione1();
}

interface interfaccia_2 {
    /* metodi astratti */
    function funzione2();
}

/* classe che implementa 2 interfacce */
class classe_test implements interfaccia_1, interfaccia_2 {
    function funzione1() {
        /* implementazione */
    }

    function funzione2() {
        /* implementazione */
    }
}
```

L'operatore “instance of”



- ✓ Viene utilizzato per verificare la classe di un oggetto

```
if($book instanceof Book)
{
    echo "Ho trovato un'istanza di Book";
} else {
    echo "Falso allarme!";
}
```

Metodi e classi final



- ✓ Se un metodo è dichiarato **final**, non può essere effettuato per esso *l'overriding* dalle classi che lo ereditano

```
class mia_classe
{
    final function mia_funzione() {
        /* questo metodo non potrà essere sovrascritto
           dalle classi che lo ereditano*/
    }
}
```

- ✓ Se una classe è dichiarata come **final**, non può essere sottoposta al meccanismo dell'ereditarietà

```
final class FinalClass
{
    /* ... */
}
```

Metodi e classi static



```
class settings
{
    static $max_users = 5;
}

echo settings::$max_users;

class lib
{
    static function get_time() {
        return time();
    }
}

echo lib::get_time();
```

Riuso del codice PHP



- ✓ L'inclusione di codice PHP in altri file avviene tramite la funzione **require(*nome_file*)**

inclusione.php

```
<?php
    echo '<p><em>Questo messaggio si trova nel file di inclusione ...</em></p>';
?>
```

main.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
        <title>Inclusione di File</title>
    </head>
    <body>
        <h1>Inclusione di File</h1>
        <?php
            echo '<p>Ecco il contenuto del file di inclusione:</p>';
            require('inclusione.php');
            echo '<p>Il messaggio precedente non era scritto in questo file!</p>';
        ?>
        </body>
    </html>
```

P
W

- ✓ Per evitare di includere un file più volte, si usa **require_once**

Require vs include



- ✓ Nelle ultime versioni del PHP non viene fatta nessuna distinzione tra l'uso di **require** e di **include**, entrambe le funzioni servono per includere file esterni e hanno la medesima sintassi
- ✓ Le due funzioni in realtà si comportano in modo diverso solo nel caso in cui ci siano degli errori di inclusione del file
 - ✓ **include** (*nome_file*) restituisce un warning, mentre lo script prosegue
 - ✓ **require** (*nome_file*) segnala l'errore e blocca lo script
- ✓ Entrambe le funzioni hanno l'estensione **_once** per evitare il caricamento multiplo dello stesso file

Altre novità in PHP 5



Gestione delle eccezioni con `try/throw/catch`

```
<?php
try {
    $a = -1; // valore inaspettato
    if ( $a < 0 )
        throw new Exception('$a è negativo');

    // il codice che segue una eccezione non viene eseguito
    echo 'Questo testo non verrà mai visualizzato';
}
catch (Exception $e) {
    echo "Eccezione intercettata: ", $e, "\n";
}
?>
```

I namespace in PHP



```
<php
namespace Acme\Database\Connection;

use Acme\Database\Connection\Connection;
use Acme\Database\Connection2\Connection as ConnectionTwo;

class MySQL
{
    // Corpo della classe

    public function getConnection()
    {
        return new Connection(); // Restituisce un'istanza della classe Acme\Dat
    }

    public function getConnection2()
    {
        return new ConnectionTwo(); // Restituisce un'istanza della classe Acme\
    }
}
```

Running example



Riprogettiamo l'esempio di riferimento utilizzando PHP e MySQL

HOME MY LIBRARY [Home](#)

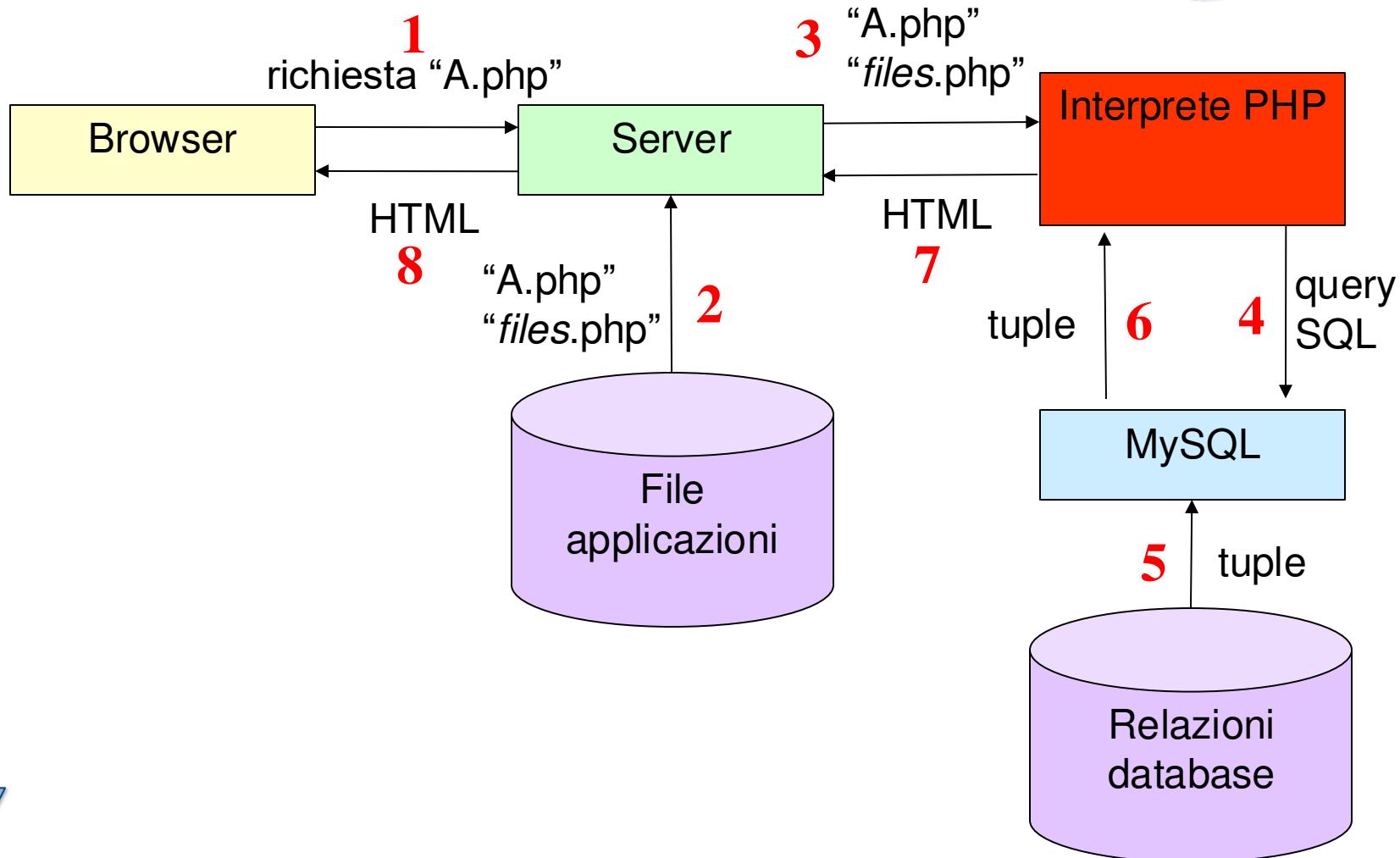
My Books List

Un semplicissimo esempio di sito web realizzato durante il corso di Programmazione Web e Servizi Digitali. Il sito riporta l'elenco dei libri che sto leggendo o che ho letto, e la lista degli autori che hanno popolato le mie letture e la mia fantasia. Il sito web continuerà a crescere durante questo semestre, completandosi di volta in volta grazie all'applicazione delle tecnologie web che verranno presentate nel corso. Buon divertimento!

Semina un atto, e raccogli un'abitudine; semina un'abitudine, e raccogli un carattere; semina un carattere, e raccogli un destino.

— [Il pensiero del Buddha]

Interazione tra PHP e MySQL



Alcune funzioni di utilità...



Per costruire porzioni ripetitive della pagina XHTML

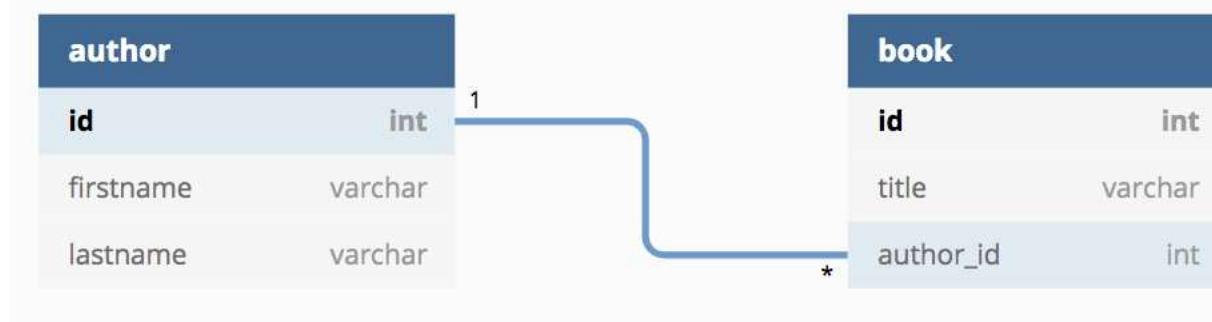
```
function html_head($titolo) {
    $result = ' <head>';
    $result .= '     <title>' . $titolo . '</title>';
    $result .= '     <meta charset="UTF-8">';
    $result .= '     <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">';
    $result .= '     <!-- Fogli di stile -->';
    $result .= '     <link rel="stylesheet" href="http://' . $_SERVER['HTTP_HOST'] . '/PW_runningExample_v3_PHP/css/bootstrap.css">';
    $result .= '     <link rel="stylesheet" href="http://' . $_SERVER['HTTP_HOST'] . '/PW_runningExample_v3_PHP/css/style.css">';
    $result .= '     <!-- jQuery e plugin JavaScript -->';
    $result .= '     <script src="http://code.jquery.com/jquery.js"></script>';
    $result .= '     <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js"></script>';
    $result .= '     <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>';
    $result .= '     <script src="http://' . $_SERVER['HTTP_HOST'] . '/PW_runningExample_v3_PHP/js/bootstrap.min.js"></script>';
    $result .= '     <!-- Bootstrap Icons -->';
    $result .= '     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css">';
    $result .= ' </head>';

    return $result;
}
```

utils/XHTML_functions.php



Il database



Alcune funzioni di utilità...



Per costruire porzioni ripetitive della pagina XHTML

Per modellare classi PHP corrispondenti alle tabelle del database
(seguendo la tecnica *ORM – Object-Relational Mapping*, che
vedremo nella programmazione agile)

- Author.php, Book.php

Per modellare il *DAO (Data Access Object)*, astraendo dalla
sottostante “storage layer”

- DataLayer.php

Alcune funzioni di utilità...



Per costruire porzioni ripetitive della pagina XHTML

Per modellare classi PHP corrispondenti alle tabelle del database
(seguendo la tecnica *ORM – Object-Relational Mapping*, che
vedremo nella programmazione agile)

- Author.php, Book.php

Per modellare il *DAO (Data Access Object)*, astraendo dalla
sottostante “storage layer”

- DataLayer.php

Connessione a MySQL



```
new PDO("mysql:host={$HOST};dbname={$DB_NAME}", $USERNAME, $PASSWORD);
```

PHP Data Object - apre una connessione al server **\$HOST**, database **\$DB_NAME**, con utente **\$USERNAME** e password **\$PASSWORD**

Esempio (I)



```
<?php  
    $USERNAME = "devis";  
    $PASSWORD = "bianchini";  
    $HOST = "localhost";  
    $DB_NAME = "MyLibraryDB";
```

utils/config.php



Esempio (II)



```
include_once('../utils/config.php');
include_once('Author.php');
include_once('Book.php');

2 references | 0 implementations
class DataLayer
{
    4 references
    private $pdo;

    /**
     * Constructor of the DataLayer class
     */
    2 references | 0 overrides
    public function __construct()
    {
        global $HOST, $USERNAME, $PASSWORD, $DB_NAME;

        try {
            $this->pdo = new PDO("mysql:host={$HOST};dbname={$DB_NAME}", $USERNAME, $PASSWORD);
            // Set the PDO error mode to exception
            $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch(PDOException $e){
            die("ERROR: Could not connect. " . $e->getMessage());
        }
    }
}
```

Esecuzione di query SQL



(new PDO) ->query (\$sql):

invia una query SQL al database a cui si è attualmente connessi

a)

```
$sql = "SELECT * FROM book ORDER BY title";
try {
    $statement = $this->pdo->query($sql);
```

b)

```
$sql = "SELECT * FROM author where id = :id";
try {
    $statement = $this->pdo->prepare($sql);
    $statement->execute(array(':id' => $id));
```

c)

```
$sql = "SELECT * FROM author where id = :id";
try {
    $statement = $this->pdo->prepare($sql);
    $statement->bindParam(':id',$id);
    $statement->execute();
```

Elaborazione del risultato (I)



`$queryStatement->fetch(PDO::FETCH_ASSOC)`:
estrae un record dal risultato di una query (ogni volta che viene invocata) sotto forma di array associativo (i nomi delle colonne del record diventano i nomi degli elementi dell' array)

```
public function findAuthorById($id)
{
    $sql = "SELECT * FROM author where id = :id";
    try {
        $statement = $this->pdo->prepare($sql);
        $statement->execute(array(':id' => $id));

        $author = $statement->fetch(PDO::FETCH_ASSOC);

        // Check if author found
        if ($author) {
            return new Author($author['id'], $author['firstname'], $author['lastname']);
        } else {
            return null; // Author not found
        }
    } catch (PDOException $e) {
        die("ERROR: Could not execute query. " . $e->getMessage());
    }
}
```

Elaborazione del risultato (I)



`$queryStatement->fetch(PDO::FETCH_ASSOC)`:
estrae un record dal risultato di una query (ogni volta che viene invocata) sotto forma di array associativo (i nomi delle colonne del record diventano i nomi degli elementi dell' array)

```
public function findAuthorById($id)
{
    $sql = "SELECT * FROM author where id = :id";
    try {
        $statement = $this->pdo->prepare($sql);
        $statement->execute(array(':id' => $id));

        $author = $statement->fetch(PDO::FETCH_ASSOC);

        // Check if author found
        if ($author) {
            return new Author($author['id'], $author['firstname'], $author['lastname']);
        } else {
            return null; // Author not found
        }
    } catch (PDOException $e) {
        die("ERROR: Could not execute query. " . $e->getMessage());
    }
}
```

\$autho	
id	2
firstname	Alessandro
lastname	Manzoni

Elaborazione del risultato (II)



`$queryStatement->fetchAll (PDO::FETCH_ASSOC)` : simile al precedente, ma in questo caso recupera tutti i record nel risultato di una query, mettendoli in un array di array associativi

```
public function listBooks()
{
    $sql = "SELECT * FROM book ORDER BY title";
    try {
        $statement = $this->pdo->query($sql);

        $books = $statement->fetchAll(PDO::FETCH_ASSOC);
        $booksList = array();
        foreach ($books as $book) {
            $author = $this->findAuthorById($book['author_id']);
            $booksList[] = new Book($book['id'], $book['title'], $author->getFirstName() . " " . $author->getLastName(), $book['author_id']);
        }
        return $booksList;
    } catch (PDOException $e) {
        die("ERROR: Could not execute query. " . $e->getMessage());
    }
}
```



Modifica del database



\$queryStatement->rowCount():

restituisce il numero di tuple coinvolte nella query appena eseguita

```
public function findBooksByAuthorID($author_id)
{
    $sql = "SELECT * FROM book where author_id = :author_id";
    try {
        $statement = $this->pdo->prepare($sql);
        $statement->bindParam(':author_id', $author_id);
        $statement->execute();

        // Get the number of affected rows
        $affectedRows = $statement->rowCount();

        if($affectedRows != 0)
        {
            return true;
        } else {
            return false;
        }
    } catch (PDOException $e) {
        die("ERROR: Could not execute query. " . $e->getMessage());
    }
}
```



Running example (I)



La lista degli autori...

Author's name

Dante Alighieri	Edit	Delete
Vito Mancuso	Edit	Delete
Alessandro Manzoni	Edit	Delete
Edgar Allan Poe	Edit	Delete



Running example (I)



La lista degli autori...

La lista è caricata automaticamente dal database...

Author's name	Edit	Delete
Dante Alighieri		
Vito Mancuso		
Alessandro Manzoni		
Edgar Allan Poe		



Running example (I)



La lista degli autori...

Author's name	Edit	Delete
Dante Alighieri		
Vito Mancuso		
Alessandro Manzoni		
Edgar Allan Poe		

The screenshot shows a web interface for managing authors. At the top, there's a navigation bar with 'HOME' and 'MY LIBRARY'. Below it, a heading 'My Authors' is followed by a table listing four authors: Dante Alighieri, Vito Mancuso, Alessandro Manzoni, and Edgar Allan Poe. Each author entry has an 'Edit' button and a 'Delete' button. A green 'Create new author' button is located in the top right. A callout bubble points to the 'Edit' button for the first author, with the text 'La lista è caricata automaticamente dal database...'. Another callout bubble points to the 'Delete' button for the second author, with the text 'Non è possibile rimuovere un autore con dei libri associati'.



Running example (II)



Creare/modificare un autore...

HOME MY LIBRARY

Home / My Library / Authors / Edit author

Edit Author

First Name Alessandro

Last Name Manzoni

Save

Cancel

Running example (II)



Creare/modificare un autore...

Una stessa pagina per creare e modificare, differenziate utilizzando `$_GET['id']`

Edit Author

First Name	<input type="text" value="Alessandro"/>
Last Name	<input type="text" value="Manzoni"/>

Running example (II)



Creare/modificare un autore...

The screenshot shows a web interface for managing authors. At the top, there's a navigation bar with a book icon, 'HOME', and 'MY LIBRARY'. Below this is a breadcrumb trail: 'Home / My Library / Authors / Edit author'. The main content area is titled 'Edit Author'. It contains two input fields: 'First Name' with 'Alessandro' and 'Last Name' with 'Manzoni'. At the bottom are two buttons: a blue 'Save' button and a red 'Cancel' button.

Una stessa pagina per creare e modificare, differenziate utilizzando `$_GET['id']`

Alcuni aspetti della pagina sono differenziati tra le due pagine (e.g., "Edit Author"/"Create new Author")



Running example (II)



Creare/modificare un autore...

HOME MY LIBRARY

Home / My Library / Authors / Edit author

Edit Author

First Name: Alessandro

Last Name: Manzoni

Save **Cancel**

Una stessa pagina per creare e modificare, differenziate utilizzando `$_GET['id']`

Alcuni aspetti della pagina sono differenziati tra le due pagine (e.g., "Edit Author"/"Create new Author")

Valori precaricati in caso di modifica di un autore (partendo dal parametro `$_GET['id']`)

Running example (II)



Creare/modificare un autore...



Una stessa pagina per creare e modificare, differenziate utilizzando `$_GET['id']`

Edit Author

First Name	<input type="text" value="Alessandro"/>
Last Name	<input type="text" value="Manzoni"/>

Save **Cancel**

Alcuni aspetti della pagina sono differenziati tra le due pagine (e.g., "Edit Author"/"Create new Author")

Valori precaricati in caso di modifica di un autore (partendo dal parametro `$_GET['id']`)

La creazione/modifica nel database è effettuata invocando lo script stesso

Running example (III)



Eliminare un autore...

HOME MY LIBRARY

Home / My Library / Authors / Delete author

Delete author "Dante Alighieri" from the list

Deleting author. Confirm?

Revert

The author **will not be removed** from the data base

Back to authors' list

Confirm

The author **will be permanently removed** from the data base

Delete

Running example (III)



Eliminare un autore...



HOME

MY LIBRARY ▾

Nome dell'autore precaricato
partendo dal parametro
`$_GET['id']`

Home / My Library / Authors / Delete author

Delete author "Dante Alighieri" from the list

Deleting author. Confirm?

Revert

The author **will not be removed** from the data base

Back to authors' list

Confirm

The author **will be permanently removed** from the data base

Delete



Running example (III)



Eliminare un autore...



HOME

MY LIBRARY ▾

Nome dell'autore precaricato
partendo dal parametro
`$_GET['id']`

Home / My Library / Authors / Delete author

Delete author "Dante Alighieri" from the list

Deleting author. Confirm?

Revert

The author **will not be removed** from the data base

Back to authors' list

Confirm

The author **will be permanently removed** from the data base

Delete

La cancellazione dal database è
effettuata invocando lo script
stesso



MANIPOLAZIONE DELL'INTESTAZIONE HTTP

Gli header HTTP



- ✓ Normalmente, PHP è usato per costruire il corpo della risposta HTTP del server, che per default presenta un'intestazione HTTP standard contenente, tra gli altri, il tipo MIME del file HTML (**text/html**)
- ✓ Talvolta è necessario gestire le intestazioni del protocollo HTTP (**headers**)
- ✓ Due tipologie di **headers**
 - ✓ **request headers**: inviati dal browser verso il server quando viene generata una richiesta HTTP (per esempio, tramite il click su un link)
 - ✓ **response headers**: inviati dal server al browser in risposta ad una determinata richiesta (per esempio, sotto forma di comune pagina HTML)

Header HTTP nella richiesta



- ✓ Per visualizzare gli headers inviati nella richiesta si può usare la funzione `getallheaders()` che produce un array contenente tutte le informazioni relative agli headers inviati in input (alias di `apache_request_headers()`)

```
<?php
# chiamata alla funzione per la raccolta dei request headers
$headers = getallheaders();
# visualizzazione dei valori dell'array tramite ciclo
foreach ($headers as $name => $content)
{
    echo "{$name} = $content<br />\n";
}
?>
```

Elenca contenuto directory

[Host] = localhost
[Connection] = keep-alive
[Accept] = text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
[User-Agent] = Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.63 Safari/537.31
[Referer] = http://localhost/prova/headers/
[Accept-Encoding] = gzip,deflate,sdch
[Accept-Language] = it-IT,it;q=0.8,en-US;q=0.6,en;q=0.4
[Accept-Charset] = ISO-8859-1,utf-8;q=0.7,*;q=0.3
[Cookie] = PHPSESSID=r7vuie4l3mi4t75f45001c5m23; mail=osor@osor.it

Header HTTP nella risposta



- ✓ Esiste un'analogia funzione `apache_response_headers()` per visualizzare gli headers nella risposta HTTP
- ✓ Talvolta è necessario modificare/aggiungere altre intestazioni
 - ✓ per re-dirigere il browser su un'altra pagina (eventualmente dopo un certo tempo)
 - ✓ per evitare il caching della pagina
 - ✓ per modificare il `content type` della pagina
 - ✓ ...
- ✓ A tale scopo esiste la funzione `header()`
- ✓ La funzione `header()` deve essere invocata prima di qualsiasi altro output, di qualunque elemento HTML o di qualunque spazio

```
# modifica dell'header relativo allo status della richiesta,
# utile nel caso in cui si debbano risolvere problematiche
# relative alla riscrittura delle URL
header($_SERVER["SERVER_PROTOCOL"]." 200 OK");

# header per le richieste non soddisfatte dal server
header($_SERVER["SERVER_PROTOCOL"]." 404 Not Found");

# header prodotto dall'impossibilità di accedere alla
# risorsa richiesta
header($_SERVER["SERVER_PROTOCOL"]." 403 Forbidden");

# header prodotto per segnalare una risorsa
# spostata in modo permanente su un percorso diverso
# da quello utilizzato per definire la richiesta
header($_SERVER["SERVER_PROTOCOL"]." 301 Moved Permanently");

# header prodotto nel caso di malfunzionamenti del server
# in seguito al tentativo di soddisfare una richiesta
header($_SERVER["SERVER_PROTOCOL"]." 500 Internal Server Error");

# rindirizzamento della richiesta su un percorso diverso
# da quello specificato
header('Location: http://www.miosito.it/');

# rindirizzamento della richiesta entro un periodo
# di tempo definito (delay)
header('Refresh: 5; url=http://www.miosito.it/');
echo 'Il browser verrà redirezionato entro 5 secondi';

# override del valore relativo all'header X-Powered-By
header('X-Powered-By: PHP/5.2.8');

# modifica dell'header relativo al linguaggio utilizzato
header('Content-language: en');

# modifica dell'header relativo alla data dell'ultima modifica
$time = time() - 60;
header('Last-Modified: '.gmdate('D, d M Y H:i:s', $time).' GMT');
```



```
# modifica dell'header relativo allo status della richiesta,
# utile nel caso in cui si debbano risolvere problematiche
# relative alla riscrittura di header
header($_SERVER["SERVER_PROTOCOL"] . " 304 Not Modified");

# header per le richieste non cacheate
header($_SERVER["SERVER_PROTOCOL"] . " 200 OK");

# header prodotto dall'impossibilità di trovare la
# risorsa richiesta
header($_SERVER["SERVER_PROTOCOL"] . " 404 Not Found");

# header prodotto per segnalare una richiesta
# spostata in modo permanente da quella utilizzata per dare
# informazioni sullo stato della richiesta
header($_SERVER["SERVER_PROTOCOL"] . " 301 Moved Permanently");

# header prodotto nel caso di un errore
# in seguito al tentativo di eseguire un comando
header($_SERVER["SERVER_PROTOCOL"] . " 500 Internal Server Error");

# rindirizzamento della richiesta
# da quello specificato
header('Location: http://www.italia.it');

# rindirizzamento della richiesta
# di tempo definito (delay)
header('Refresh: 5; url=http://www.italia.it');

echo 'Il browser verrà redirettato verso il sito italiano';

# override del valore relativo alla data dell'ultima modifica
header('X-Powered-By: PHP/5.2.14');

# modifica dell'header relativo alla data dell'ultima modifica
header('Content-language: en-US');

# modifica dell'header relativo alla data dell'ultima modifica
$time = time() - 60;
header('Last-Modified: ' . gmdate('D, d M Y H:i:s', $time) . ' GMT');

# header per segnalare l'assenza di modifiche
header($_SERVER["SERVER_PROTOCOL"] . " 304 Not Modified");

# impostazione del Content-Length per operazioni di caching
header('Content-Length: 168');

# header per il download
header('Content-Type: application/octet-stream');
header('Content-Disposition: attachment; filename="file.zip"');
header('Content-Transfer-Encoding: binary');

# scaricamento di un file
readfile('file.zip');

# disabilitazione della cache per il documento corrente
header('Cache-Control: no-cache, no-store, max-age=0, must-revalidate');
header('Expires: Mon, 26 Jul 1997 05:00:00 GMT'); # Date in the past
header('Pragma: no-cache');

# impostazione del content type per
# le varie tipologie di estensioni:
header('Content-Type: text/html; charset=iso-8859-1');
header('Content-Type: text/html; charset=utf-8');
header('Content-Type: text/plain');
header('Content-Type: image/jpeg');
header('Content-Type: application/zip');
header('Content-Type: application/pdf');
header('Content-Type: audio/mpeg');
header('Content-Type: application/x-shockwave-flash');

# richiesta di autenticazione
header($_SERVER["SERVER_PROTOCOL"] . " 401 Unauthorized");
header('WWW-Authenticate: Basic realm="Accesso Riservato"');
echo 'Inserire i dati per il login';
```

Buffering



- ✓ Le intestazioni devono precedere qualunque contenuto, perché una volta che la pagina inizia ad essere inviata non è più possibile modificare l'intestazione della risposta HTTP

- ✓ Questo vincolo può essere aggirato con il meccanismo del *buffering*
 - ✓ la funzione `ob_start()` viene posta immediatamente prima del contenuto della pagina, compresi eventuali spazi bianchi
 - ✓ in tal modo, lo script non inizia ad inviare la pagina fino a quando non ha terminato la sua esecuzione
 - ✓ si può ripristinare il consueto meccanismo di emissione immediata dell'output invocando la funzione `ob_end_flush()`



PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - PHP

Prof. Ada Bagozi
ada.bagozi@unibs.it





PROGRAMMAZIONE WEB

PHP APPROFONDIMENTI

ARRAY E STRINGHE

Prof. Ada Bagozi

ada.bagozi@unibs.it



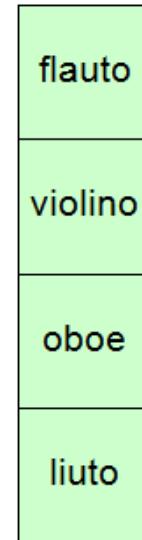
Array



Array = variabile che contiene un insieme (indicizzato) di **elementi**

- ✓ Elemento: scalare (semplice) o un altro array
- ✓ Individuazione degli elementi mediante indicizzazione
 - **numerica**
 - **associativa**

`$strumenti`



Array indicizzati numericamente



Creazione di array mediante enumerazione dei suoi elementi:

```
$strumenti = array('flauto', 'violino', 'oboe', 'liuto');
```

Creazione di array mediante assegnamento con un altro array:

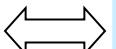
```
$a = $strumenti;
```

```
$numeri = range(1, 10);
```



```
$numeri = array(1,2,3,4,5,6,7,8,9,10);
```

```
$lettere = range('a', 'z');
```



```
$lettere = array('a','b', ..., 'z');
```



Creazione di array mediante caricamento da database

Array associativi



Indice rappresentato da un nome (**chiave**):

```
$prezzi = array ('flauto'=>2500,  
                  'violino'=>8000,  
                  'oboe'=>5400,  
                  'liuto'=>11000);
```

flauto	2500
violino	8000
oboe	5400
liuto	11000

Operatory per array



Simbolo	Nome	Esempio	Risultato
+	Unione	<code>\$a + \$b</code>	Concatenazione di <code>\$a</code> con gli elementi di <code>\$b</code> che hanno indici <u>diversi</u> da quelli in <code>\$a</code>
==	Uguaglianza	<code>\$a == \$b</code>	Uguaglianza di <code>\$a</code> e <code>\$b</code>
===	Identità	<code>\$a === \$b</code>	Identità di <code>\$a</code> e <code>\$b</code>
!=	Disuguaglianza	<code>\$a != \$b</code>	Disuguaglianza di <code>\$a</code> e <code>\$b</code>
!==	Non identità	<code>\$a !== \$b</code>	Non identità di <code>\$a</code> e <code>\$b</code>

P
W

```
$a = array(1,2,3);  
$b = array(4,5,6,7,8);  
$c = array('1','2','3');
```



```
$a + $b: (1,2,3,7,8)  
$a == $b: false  
$a == $c: true  
$a === $c: false
```

Array multidimensionali (I)



Indicizzazione numerica:

```
$articoli = array(array('FLT', 'flauto', 2500),  
                  array('VNL', 'violino', 8000),  
                  array('OBO', 'oboe', 5400),  
                  array('LUT', 'liuto', 11000));
```

FLT	flauto	2500
VNL	violino	8000
OBO	oboe	5400
LUT	liuto	11000

Array multidimensionali (II)



Accesso mediante ciclo **for**:

```
for($riga=0; $riga<4; $riga++)  
{  
    for($colonna=0; $colonna<3; $colonna++)  
        echo $articoli[$riga][$colonna] . ' '  
    echo '<br/>';  
}
```

↓
FLT flauto 2500
VNL violino 8000
OBO oboe 5400
LUT liuto 11000

FLT	flauto	2500
VNL	violino	8000
OBO	oboe	5400
LUT	liuto	11000

Array multidimensionali (III)



Indicizzazione mista:

```
$articoli = array(array('codice'=>'FLT',
                         'strumento'=>'flauto',
                         'prezzo'=>2500),
                  array('codice'=>'VLN',
                         'strumento'=> 'violino',
                         'prezzo'=>8000),
                  array('codice'=>'OBO',
                         'strumento'=>'oboe',
                         'prezzo'=>5400),
                  array('codice'=>'LUT',
                         'strumento'=>'liuto',
                         'prezzo'=> 11000));
```

codice strumento prezzo

FLT	flauto	2500
VLN	violino	8000
OBO	oboe	5400
LUT	liuto	11000



Array multidimensionali (IV)



Accesso mediante ciclo **foreach**:

```
foreach($articoli as $articolo)
{
    foreach($articolo as $indice => $contenuto)
        echo "$indice: $contenuto ";
    echo '<br/>';
}
```

\$articoli

codice strumento prezzo

FLT	flauto	2500
VNL	violino	8000
OBO	oboe	5400
LUT	liuto	11000



\$articolo

```
codice: FLT strumento: flauto prezzo: 2500
codice: VLN strumento: violino prezzo: 8000
codice: OBO strumento: oboe prezzo: 5400
codice: LUT strumento: liuto prezzo: 11000
```

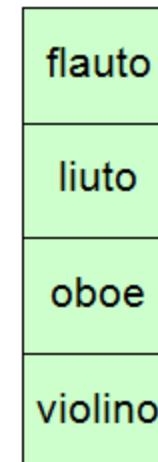
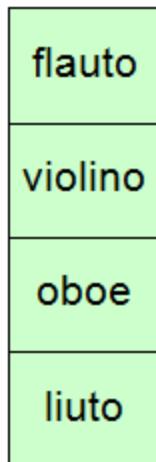


Ordinamento di un array (I)



Ordinamento di array indicizzati numericamente: **sort()**

```
$strumenti = array('flauto', 'violino', 'oboe', 'liuto');  
sort($strumenti);
```



Ordinamento di un array (II)



Ordinamento di array indicizzati associativamente: **ksort()** e **asort()**

```
$prezzi = array('flauto'=>2500,  
                 'violino'=>8000,  
                 'oboe'=>5400,  
                 'liuto'=>11000);
```



flauto	2500
violino	8000
oboe	5400
liuto	11000

```
ksort($prezzi);
```



flauto	2500
liuto	11000
oboe	5400
violino	8000

```
asort($prezzi);
```



flauto	2500
oboe	5400
violino	8000
liuto	11000

Ordinamento di un array (III)



Ordinamento di array in ordine inverso: **rsort()**, **krsort()** e **arsort()**

```
rsort($strumenti);
```

flauto
violino
oboe
liuto



violino
oboe
liuto
flauto

```
krsort($prezzi);
```

flauto	2500
violino	8000
oboe	5400
liuto	11000



violino	8000
oboe	5400
liuto	11000
flauto	2500

```
arsort($prezzi);
```

flauto	2500
violino	8000
oboe	5400
liuto	11000



liuto	11000
violino	8000
oboe	5400
flauto	2500

Ordinamento casuale di un array



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Prova</title>
  </head>
  <body>
    <?php
      $a = array(1,2,3);
      $b = array(4,5,6,7,8);
      $c = $a + $b;

      shuffle($c);

      for($i=0; $i<count($c); $i++)
      {
        echo $c[$i] . "<br>";
      }
    ?>
    </body>
</html>
```

Inversione di un array



array_reverse(\$a): genera un array inverso di **\$a**

```
$inv_str = array_reverse($strumenti);
```

flauto
violino
oboe
liuto



liuto
oboe
violino
flauto

```
$inv_art = array_reverse($articoli);
```

codice strumento prezzo

FLT	flauto	2500
VNL	violino	8000
OBO	oboe	5400
LUT	liuto	11000



codice strumento prezzo

LUT	liuto	11000
OBO	oboe	5400
VNL	violino	8000
FLT	flauto	2500

Conteggio degli elementi di un array



count (\$a) : restituisce il numero di elementi dell'array **\$a**

array_count_values (\$a) : restituisce un array associativo che indica la frequenza di ogni elemento nell'array **\$a** (gli elementi di **\$a** devono essere scalari)

flauto
violino
flauto
flauto
violino
oboe
liuto
liuto



flauto	3
violino	2
oboe	1
liuto	2

Da array associativi a variabili scalari



extract(array *a* [, int *tipologia* [, string *prefisso*]]): genera un insieme di variabili omonime delle chiavi di *a*

```
extract($prezzi);  
echo "$flauto $violino $oboe $liuto"
```

flauto	2500
violino	8000
oboe	5400
liuto	11000

2500 8000 5400 11000

Parametri opzionali:

- ✓ ***tipologia***: tipologia di gestione delle collisioni (default: sovrascrittura)
- ✓ ***prefisso***: rilevante per certe tipologie di collisione

tipologia	Significato
EXTR_OVERWRITE	Sovrascrive le variabili in caso di collisione
EXTR_SKIP	Salta l'elemento che provoca la collisione
EXTR_PREFIX_SAME	Crea una variabile \$prefisso_chiave solo in caso di collisione
EXTR_PREFIX_ALL	Crea tutte le variabili \$prefisso_chiave , indipendentemente dalle collisioni
EXTR_PREFIX_INVALID	Crea una variabile \$prefisso_chiave in caso di identificatore (chiave) non valido
EXTR_IF_EXISTS	Estrae solo le variabili che già esistono

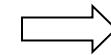
Navigazione di un array



Array: ha un puntatore interno che punta all'elemento corrente dell'array (creazione array → punta al primo elemento)

- ✓ **current(\$a)** → restituisce elemento corrente
- ✓ **each(\$a)** → restituisce elemento corrente; incrementa puntatore
- ✓ **next(\$a)** → incrementa puntatore; restituisce elemento corrente
- ✓ **reset(\$a)** → posiziona puntatore al primo elemento (e lo restituisce)
- ✓ **end(\$a)** → posiziona puntatore all'ultimo elemento (e lo restituisce)
- ✓ **prev(\$a)** → decrementa puntatore; restituisce elemento corrente

```
$a = array("alfa", "beta", "gamma");
$stringa = end($a);
while($stringa)
{
    echo "$stringa <br/>";
    $stringa = prev($a);
}
```



gamma
beta
alfa

Pulitura di stringhe



string trim(string stringa): rimozione di spaziatura intorno a *stringa*
Spazio bianco ''

Newline "\n"

Carriage return "\r"

Tab orizzontale "\t"

Tab verticale "\x0B"

Fine stringa "\0"

```
$nome = trim($_POST['nome']);  
$email = trim($_POST['email']);
```

string ltrim(string stringa): pulitura solo a sinistra

string rtrim(string stringa): pulitura solo a destra

Rendering dei Newline



string nl2br(string stringa): sostituzione di "\n" con "
"

Strumenti Musicali in Franciacorta - Feedback - Windows Internet Explorer

http://127.0.0.1/php-stringhe/gestisci_feedback.php

Google Effettua la ricerca

Preferiti Siti suggeriti Customize Links Raccolta Web Slice

Strumenti Musicali in Franciacorta - Feedback

Strumenti Musicali in Franciacorta

Invio Feedback

Luigi, il tuo feedback è stato spedito.

Senza nl2br():

Nome cliente: Luigi Email cliente: Luigi Feedback cliente: Ho trovato il negozio ben fornito. Gli strumenti musicali sono 'speciali'. Forse, è solo 'un pò caro' ...

Con nl2br():

Nome cliente: Luigi
Email cliente: Luigi
Feedback cliente:
Ho trovato il negozio ben fornito.
Gli strumenti musicali sono 'speciali'.
Forse, è solo 'un pò caro' ...

Intranet locale | Modalità protetta: disattivata

Formattazione di stringhe



```
void printf(string formato [, mixed args ...])  
string sprintf(string formato [, mixed args ...])
```

formato: specifica del formato della stringa di output mediante codici
args: variabili che istanziano i codici nel **formato**

```
echo "Il costo totale è $costo_totale";
```



```
Il costo totale è 83,8
```

```
printf("Il costo totale è %s", $costo_totale);
```

```
printf("Il costo totale è %.2f", $costo_totale);
```

```
Il costo totale è 83,80
```

Specifiche di conversione multiple:

```
printf("Il costo totale è %.2f (con spedizione %.2f)",  
      $costo_totale, $costo_con_spedizione);
```

```
Il costo totale è 83.80 (con spedizione 88.80)
```

Formattazione di stringhe



Sintassi specifica di conversione:

`%['carattere-riempimento'] [-] [larghezza] [.precisione] codice`

codice	Interpretazione	Stampa
b	Integer	Numero binario
c	Integer	Carattere
d	Integer	Numero decimale
f	Double	Numero in virgola mobile
o	Integer	Numero ottale
s	String	Stringa
u	Integer	Intero senza segno
x	Intero	Numero esadecimale con caratteri minuscoli (a-f)
X	Intero	Numero esadecimale con caratteri maiuscoli (A-F)

Formattazione di stringhe



Cambiamento del **case** (maiuscola-minuscola) di stringhe:

```
$subject = 'Feedback dal sito web';
```

Funzione	Effetto	Output
strtoupper (\$subject)	Uppercase	FEEDBACK DAL SITO WEB
strtolower (\$subject)	Lowercase	feedback dal sito web
ucfirst (\$subject)	Maiuscola solo del primo carattere della stringa se alfabetico	Feedback dal sito web
ucwords (\$subject)	Maiuscola del primo carattere di ogni parola della stringa se alfabetico	Feedback Dal Sito Web

Utile nel confronto tra stringhe:

```
if(strtolower($email) == 'luigi.rossi@alice.it')  
    $to = 'marketing@strumentifranciacorta.com';  
else  
    $to = 'info@strumentifranciacorta.com';
```

Formattazione di stringhe



Problemi nella memorizzazione di stringhe nel database: apici, \, NULL
Inibizione dei metacaratteri mediante backslash: \', \\, \\\

```
string addslashes(string stringa)  
string stripslashes(string stringa)
```

```
$feedback = addslashes(trim($_POST['feedback']));  
$feedback = stripslashes($feedback);
```

The screenshot shows a Windows Internet Explorer window with the following content:

```
Strumenti Musicali in Franciacorta - Feedback - Windows Internet Explorer
http://127.0.0.1/php-stringhe/gestisci_feedback.php

Feedback:  
Ho trovato il negozio ben fornito. Gli strumenti musicali sono 'speciali'. Forse, è solo 'un pò caro' ...  
  
Feedback dopo addslashes():  
Ho trovato il negozio ben fornito. Gli strumenti musicali sono \speciali\. Forse, è solo \un pò caro\ ...  
  
Feedback dopo stripslashes():  
Ho trovato il negozio ben fornito. Gli strumenti musicali sono 'speciali'. Forse, è solo 'un pò caro' ...
```

Esplosione ed implosione di stringhe



```
array explode(string separatore, string input [, int limite])
```

```
$email_array = explode('@', $email);
if(strtolower($email_array[0] == 'luigi.rossi')
    $to = 'marketing@strumentifranciacorta.com';
else
    $to = info@strumentifranciacorta.com';
```

```
string implode(array input, string separatore)
```

```
$email = implode($email_array, '@');
```

string strtok(string input, string separatore): preleva da *input* un token alla volta

```
$token = strtok($feedback, ' ');
do{
    echo "$token <br />";
    $token = strtok(' ');
} while ($token != '');
```

← Prima call → due parametri

← Successive call → un solo parametro (separatore)

Prelievo di sottostringhe



string substr(string stringa, int inizio [, int lung])

inizio: indice da cui prelevare la sottostringa (se negativo → dalla fine)

lung: numero di caratteri da prelevare (se non specificato → fino alla fine)

```
$feedback = 'Il vostro servizio clienti è eccellente';
```

Call	Output
substr(\$feedback, 3)	vostro servizio clienti è eccellente
substr(\$feedback, 3, 6)	vostro
substr(\$feedback, -20)	clienti è eccellente
substr(\$feedback, -20, 7)	clienti

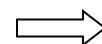
Confronto tra stringhe



```
int strcmp(string stringa1, string stringa2)
    stringa1 == stringa2 → risultato = 0
    stringa1 > stringa2 → risultato > 0
    stringa1 < stringa2 → risultato < 0
```

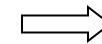
int strnatcmp(string stringa1, string stringa2):
confronto sulla base di un criterio 'naturale'
(<http://www.naturalordersort.org>)

```
echo strcmp('15', '2');
```



-1

```
echo strnatcmp('15', '2');
```



1

Versioni non sensibili alle maiuscole/minuscole (*case insensitive*):

strcasecmp() **strnatcasecmp()**

Ricerche di sottostringhe



string strstr(string pagliaio, string ago)

se **ago** trovato nel **pagliaio** → sottostringa dall'inizio di **ago**

se **ago** non trovato nel **pagliaio** → **false**

se più occorrenze di **ago** nel **pagliaio** → prima occorrenza di **ago**

Assegnamento del destinatario in base a parole chiave nel feedback:

```
$to = 'feedback@strumentifranciacorta.com'; // default
if(strstr($feedback, 'negozi'))
    $to = 'vendite@strumentifranciacorta.com';
elseif(strstr($feedback, 'consegna'))
    $to = 'spedizioni@strumentifranciacorta.com';
elseif(strstr($feedback, 'fattura'))
    $to = 'amministrazione@strumentifranciacorta.com';
```

Varianti:

stristr(): case insensitive

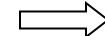
strrchr() → sottostringa partendo dall'ultima occorrenza di **ago**

Posizione di sottostringhe



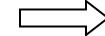
int strpos(string pagliaio, string ago [, int offset]) →
restituisce la posizione della prima occorrenza di **ago** in **pagliaio**
offset: punto in **pagliaio** da cui iniziare a cercare
se **ago** non si trova nel **pagliaio** → **false**
più veloce di **strstr()**

```
$saluto = 'Hello world';
echo strpos($saluto, 'world');
```



6

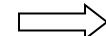
```
echo strpos($saluto, 'l', 5);
```



9

Se **ago** non si trova nel **pagliaio** → **false**: **strpos()** rimane non definito

```
$pos = strpos($saluto, 'Hello');
if($pos === false)
    echo 'Non trovato';
else
    echo "Trovato alla posizione $pos";
```



Trovato alla
posizione 0



Sostituzione di sottostringhe (I)



```
mixed str_replace(mixed ago, mixed nuovo, mixed pagliaio  
[ , int &tot ])
```

sostituisce tutte le occorrenze di *ago* in *pagliaio*

restituisce il nuovo *pagliaio*

se specificato, assegna a *tot* il numero di sostituzioni effettuate

```
$saluto = 'Hello world, wonderful world!';  
$tot = 0;  
echo str_replace('world', 'people', $saluto, $tot)."($tot)";
```

Possibile passare array invece di stringhe:

```
$feedback = 'Questo negozio fa schifo! Veramente schifo!';  
$censurate = array('schifo', ...);  
echo str_replace($censurate, '***', $feedback);
```

Sostituzione di sottostringhe (I)



```
mixed str_replace(mixed ago, mixed nuovo, mixed pagliaio  
[ , int &tot ])
```

sostituisce tutte le occorrenze di *ago* in *pagliaio*

restituisce il nuovo *pagliaio*

se specificato, assegna a *tot* il numero di sostituzioni effettuate

```
$saluto = 'Hello world, wonderful world!';  
$tot = 0;  
echo str_replace('world', 'people', $saluto, $tot)."($tot)";
```

```
Hello people, wonderful people! (2)
```

Possibile passare array invece di stringhe:

```
$feedback = 'Questo negozio fa schifo! Veramente schifo!';  
$censurate = array('schifo', ...);  
echo str_replace($censurate, '***', $feedback);
```

```
Questo negozio fa ***! Veramente ***!
```

Sostituzione di sottostringhe (III)



```
string substr_replace(string stringa, string sostituto,  
                     int inizio [ , int lung ])
```

sostituisce una parte di *stringa* con *sostituto*

inizio: posizione di *stringa* in cui inizia la sostituzione (se negativo, posizione dalla fine)

lung positiva o nulla → numero di caratteri sostituiti

lung negativa → posizione (dalla fine) su cui terminare la sostituzione

```
$saluto = 'Hello world';  
echo substr_replace($saluto, 'people', 6, 5);
```

Hello people

```
echo substr_replace($saluto, 'people ', 6, 0);
```

Hello people world

```
echo substr_replace($saluto, 'people ', 6, -2);
```

Hello people ld

W

Espressioni regolari



Pattern matching complessi (non semplicemente uguaglianze di stringhe)

Due possibili stili: POSIX, Perl (PCRE)

- ✓ **Espressione regolare** = notazione per specificare un pattern in un testo
- ✓ Uso di **metacaratteri** per specificare particolari pattern

Insiemi di caratteri (I)



Un qualsiasi carattere diverso da \n: .

.emo

→ compatibile con remo, temo, memo, ...

Classe di caratteri: [*lista-di-caratteri*]

[aeiou]

→ vocali

[a-z]

→ range delle lettere alfabetiche minuscole

[a-zA-Z]

→ range delle lettere alfabetiche

[^a-z]

→ qualsiasi carattere che non sia una lettera alfabetica minuscola

[^aeiou]

→ qualsiasi carattere che non sia una lettera vocale

Insiemi di caratteri (II)



Classi di caratteri **predefinite**:

Classe	Matches	Espansione
<code>[[:alnum:]]</code>	Caratteri alfanumerici	<code>[0-9a-zA-Z]</code>
<code>[[:alpha:]]</code>	Caratteri alfabetici	<code>[a-zA-Z]</code>
<code>[[:ascii:]]</code>	Caratteri ASCII (7 bit)	<code>[\x01-\x7F]</code>
<code>[[:blank:]]</code>	Spaziatura orizzontale	<code>[\t]</code>
<code>[[:cntrl:]]</code>	Caratteri di controllo	<code>[\x01-\x1F]</code>
<code>[[:digit:]]</code>	Cifre decimali	<code>[0-9]</code>
<code>[[:graph:]]</code>	Caratteri stampabili con inchiostro	<code>[^\x01-\x20]</code>
<code>[[:lower:]]</code>	Lettere minuscole	<code>[a-z]</code>
<code>[[:print:]]</code>	Caratteri stampabili (<code>graph</code> , spazio e <code>\t</code>)	<code>[\t\x20-\xFF]</code>
<code>[[:punct:]]</code>	Punteggiatura	<code>[-!"#\$%&'()*+,.:/;<=>?@[\\\]\^_`{ }~]</code>
<code>[[:space:]]</code>	Caratteri bianchi (incluso <code>\x0B</code> = tab verticale)	<code>[\n\r\t\x0B]</code>
<code>[[:upper:]]</code>	Lettere maiuscole	<code>[A-Z]</code>
<code>[[:xdigit:]]</code>	Cifre esadecimali	<code>[0-9a-fA-F]</code>

Ripetizione e alternativa



Ripetizione zero o più volte del pattern: *

[aeiou] *

→ Zero o più vocali

[:alpha:] [:alnum:] *

→ Lettera seguita da zero o più caratteri alfanumerici (identificatore)

Ripetizione una o più volte del pattern: +

[:digit:] +

→ Una o più cifre decimali (numero intero)

[:digit:] +, [:digit:] +

→ Numero reale con parte decimale

Opzionalità: ?

[:digit:] + [aeiou] ?

→ Numero eventualmente seguito da una vocale

Alternativa: |

org | net | com

Sottoespressioni



Possibile raggruppare sottoespressioni mediante parentesi:

```
(molto ) * grande
```

```
grande  
molto grande  
molto molto grande ...
```

Ripetizione di un pattern un numero vincolato di volte: { *limiti* }

```
[ [ :digit: ] ] {16}
```

Numero intero di 16 cifre decimali

```
(molto ) {1,3} grande
```

```
molto grande  
molto molto grande  
molto molto molto grande
```

```
(molto ) {2,} grande
```

```
molto molto grande  
molto molto molto grande  
molto molto molto molto grande
```

Ancoraggio



Possibile specificare la posizione di una sottostringa:

^Hello



Hello world
Hello people
Hello everybody

world\$



Hello world
Beutiful world
Best world

^ [[:alpha:]] [[:alnum:]] * \$



Stringa costituita unicamente
da un identificatore

[[:<:]] [[:digit:]]+ [[:>:]]



123 alfa25 460

P
W

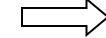
Caratteri speciali



Per inibire i caratteri speciali al di fuori di [], è necessario precederli con \

Meglio inserire il pattern tra apici singoli (non doppi):

```
'[:digit:]]+\.\[:digit:]]+'
```



```
12.0  
0.14  
13.156
```

Se pattern tra apici doppi → parsing della stringa → necessari ulteriori \

```
"[:digit:]]+\.\[:digit:]]+"
```

Modulo di feedback



Rilevazione di particolari termini nel feedback del cliente:

negozi|consegna|fattura

Controllo nome del cliente:

Luigi Rossi

G. Luigi Rossi

G.C. Luigi Rossi

`^(([[:upper:]]\.)+)?[:upper:][:lower:]+ [:upper:][:lower:]+$`

Controllo formato indirizzo di email:

Luigi.Rossi-89@domain-191.unisv.it

`^ [a-zA-Z0-9_.-]+@[a-zA-Z0-9-]+\.\[a-zA-Z0-9_.-]+\$`



Sottostringhe ed espressioni regolari



bool ereg(string pattern, string stringa)

- ✓ stabilisce se *stringa* contiene una sottostringa compatibile con *pattern*
- ✓ case-sensitive

Variante case-insensitive: **eregi()**

```
if(!eregi('^[a-zA-Z0-9_.-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9_.-]+$', $email))
{
    echo '<p>Indirizzo di email non valido: torna alla pagina precedente</p>';
    exit;
}
$to = 'info@strumentifranciacorta.com';
if(eregi('negozi|servizi', $feedback))
    $to = 'vendite@strumentifranciacorta.com';
elseif(eregi('spedizioni|adempimento', $feedback))
    $to = 'spedizioni@strumentifranciacorta.com';
elseif(eregi('conto|biglietto|fattura', $feedback))
    $to = 'amministrazione@strumentifranciacorta.com';
if(eregi('luigi.rossi@alice.it', $email))
    $to = 'marketing@strumentifranciacorta.com';
```

Sottostringhe ed espressioni regolari



```
string ereg_replace(string pattern, string sostituto, string  
                     stringa)
```

- ✓ sostituisce con *sostituto* tutte le sottostringhe di *stringa* compatibili con *pattern*
- ✓ restituisce il risultato della sostituzione (funzionale: *stringa* non cambia!)

Variante case-insensitive: *ereg_i_replace()*

```
$transazione = 'Carta di credito: 4367-2234-1245-3200';  
  
$pattern = '[[[:digit:]]{4}(-[[[:digit:]]{4}){3}';  
  
echo ereg_replace($pattern, 'xxxx-xxxx-xxxx-xxxx', $transazione);
```

Carta di credito: xxxx-xxxx-xxxx-xxxx

```
$stringa = 'alfa 23beta gamma48 125';  
  
$pattern = '[[[:<:]][:digit:]]+[[:>:]]';  
  
echo ereg_replace($pattern, 'NUM', $stringa);
```

alfa 23beta gamma48 NUM

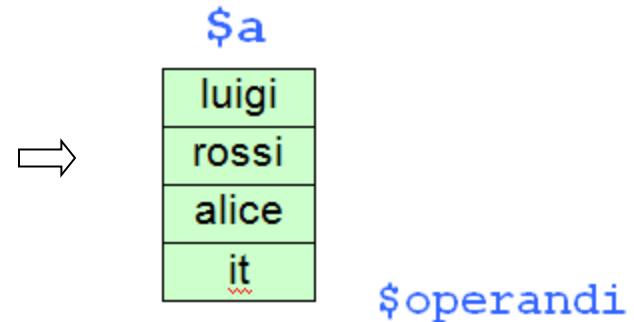
Sottostringhe ed espressioni regolari



array split(string pattern, string stringa [, int max])

- ✓ restituisce l'array di stringhe ottenuto dividendo *stringa* limitata dai confini compatibili con *pattern*
- ✓ *max*: numero massimo di elementi nell'array risultante

```
$email = 'luigi.rossi@alice.it';
$a = split('.|@', $email);
```



```
$espressione = '3*52+24/5-12';
$operandi = split('[-+*/]', $espressione);
```



Funzioni *deprecated*



`ereg('pattern','stringa')` → sostituito da
`preg_match('/pattern/','stringa')`

`eregi('pattern','stringa')` → sostituito da
`preg_match('/pattern/i','stringa')`

`ereg_replace('pattern','sostituto','stringa')` → sostituito da
`preg_replace('/pattern/','sostituto','stringa')`
uso di `'/pattern/i'` in caso di **`eregi_replace`**

`split('pattern','stringa')` → sostituito da
`preg_split('/pattern/','stringa')`



PROGRAMMAZIONE WEB E SERVIZI DIGITALI

PHP APPROFONDIMENTI

ARRAY E STRINGHE

Prof. Ada Bagozi

ada.bagozi@unibs.it





PROGRAMMAZIONE WEB

**PROGRAMMAZIONE AGILE
CON LARAVEL**

Prof. Ada Bagozi

ada.bagozi@unibs.it

**P
W**

I pilastri dello sviluppo agile



- ✓ Limitare il più possibile la scrittura di codice ripetitivo
 - ✓ e.g., utilizzo di ORM (Object-Relational Mapping)
 - ✓ e.g., generazione automatica di codice ripetitivo
- ✓ Sviluppare facendo uso di Design Pattern
- ✓ Creare gruppi di lavoro in cui fin dall'inizio gli sviluppatori SW lavorano fianco a fianco con gli end-user
 - ✓ Focus sulla prototipazione rapida

Cos'è un framework di sviluppo?



- ✓ Un **framework** è una struttura logica a supporto dello sviluppo software
- ✓ Quasi sempre un framework implementa un particolare **design pattern**
- ✓ Composto da una serie di librerie di codice in uno specifico linguaggio di programmazione
- ✓ Talvolta accompagnato da strumenti di sviluppo, come IDE o debugger
- ✓ Un framework presuppone l'adozione di una specifica **metodologia di programmazione**

Laravel



Un ***web application framework*** open source e free,
creato da Taylor Otwell

Orientato allo sviluppo agile di applicazioni Web in
accordo con il ***design pattern MVC***

A partire da Marzo 2015, uno dei più importanti
framework PHP (altri esempi, Symfony2, Nette,
CodeIgniter, Yii2)



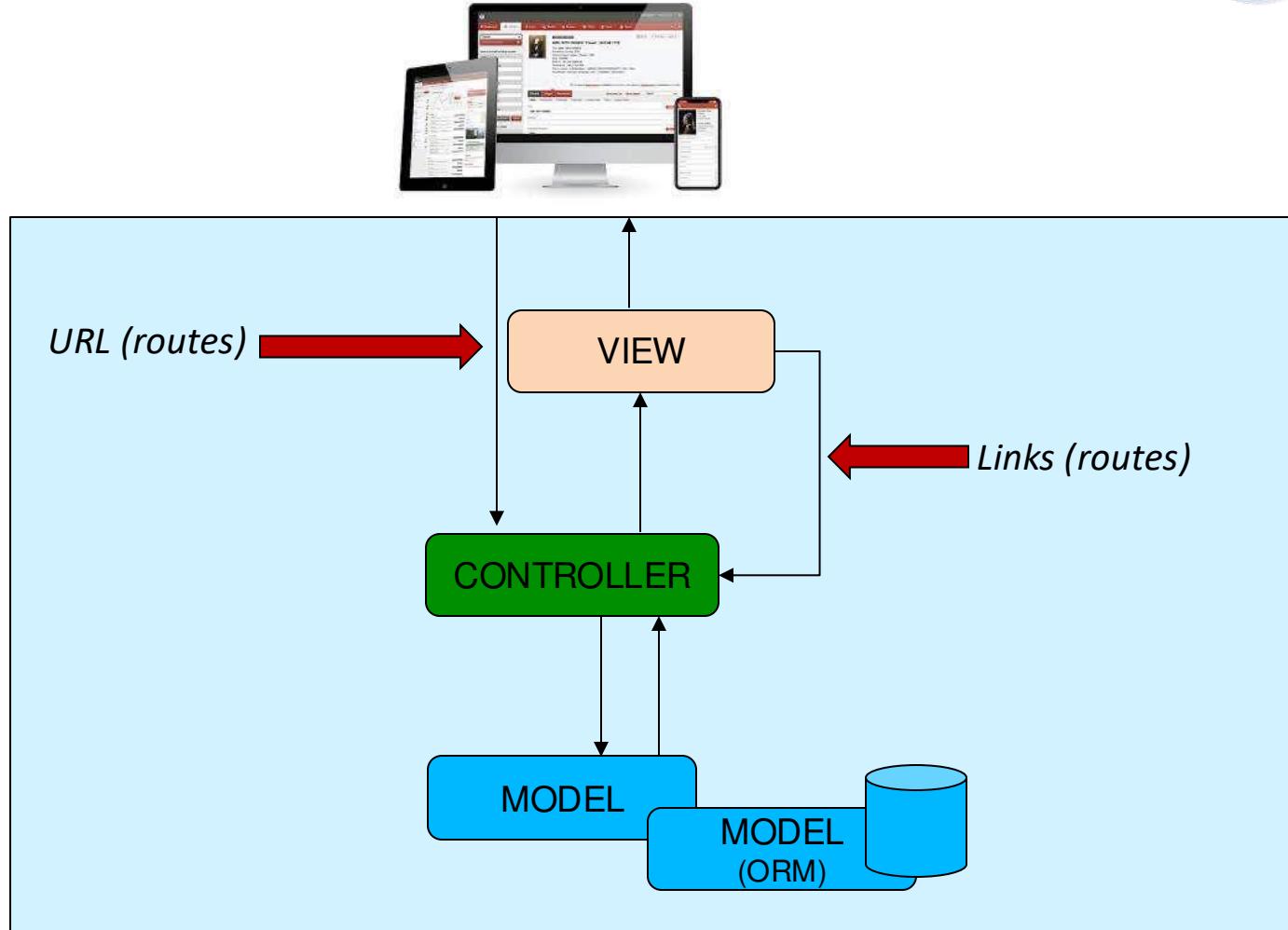
MVC design pattern (I)



Model-View-Controller (MVC) è un design pattern architetturale che semplifica lo sviluppo di applicazioni separando i componenti software in

- ✓ **Model**: componenti che forniscono i metodi per accedere ed elaborare i dati utili all'applicazione
- ✓ **View**: componenti che visualizzano i dati contenuti nel Model e si occupano dell'interazione con utenti e agenti
- ✓ **Controller**: componenti che ricevono i comandi dall'utente (attraverso le View) e li attuano invocando gli altri componenti (Model)

MVC design pattern (II)



Laravel – Requisiti



Versione corrente 12 (Febbraio 2025)

Il framework **Laravel** presenta alcuni (non molti) requisiti (<https://laravel.com/>); per la versione 12:

- ✓ PHP \geq 8.2 (php -v oppure php --version)
- ✓ Alcune estensioni PHP

Laravel utilizza **Composer** (<https://getcomposer.org/>) per la gestione delle dipendenze. Di conseguenza, prima di utilizzare Laravel, è necessario assicurarsi che Composer sia installato sulla macchina

Laravel – Versioni



Versione	Data di rilascio	Versione PHP	Bug fix fino al	Security fix fino al
1.0	Giugno 2011			
2.0	Settembre 2011			
3.0	22 Febbraio 2012			
3.1	27 Marzo 2012			
3.2	22 Maggio 2012			

...

7	3 Marzo 2020 ^[13]	≥ 7.2.5	6 Ottobre 2020	3 Marzo 2021
8	8 Settembre 2020 ^[14]	≥ 7.3	26 Luglio 2022	24 Gennaio 2023
9	8 Febbraio 2022 ^[15]	≥ 8.0 ^[16]	8 Agosto 2024	6 Febbraio 2024
10	7 Febbraio 2023 ^[15]	≥ 8.1 ^[17]	6 Agosto 2024	4 Febbraio 2025
11	12 Marzo 2024 ^[18]	≥ 8.2 ^[19]	5 Agosto 2025	3 Febbraio 2026
12 ^[20]	Q1 2025	≥ 8.2	Q3 2026	Q1 2027

Legenda Vecchia versione, nessun supporto Vecchia versione, ancora supportata Versione corrente Versione futura



Creazione di un nuovo progetto



Tramite l'utilizzo di **Composer**:

- ✓ creando il progetto direttamente, utilizzando le librerie Laravel scaricate dalla rete

```
composer create-project laravel/laravel:^12.0 ${applicationName}
```

Per scaricare e aggiornare tutte le dipendenze:

```
composer update
```

Per far partire l'applicazione:

```
php artisan serve
```

start a local server on 8000 port



Struttura delle directory (I)



app/ - contiene la business logic dell'applicazione, raccoglie tra gli altri i file relativi ai controller

(**app/http/Controllers/**) e le classi del Model
(**app/Models/**)

routes/ - contiene i file relativi alle rotte, ovvero la gestione degli URL e i mapping tra questi e le funzionalità offerte dall'applicazione

config/ - raccoglie i file contenenti le impostazioni di configurazione dell'intera applicazione

public/ - contiene tutte le risorse statiche pubbliche (script JS, CSS, immagini, font, etc.) del progetto

Struttura delle directory (II)



resources/ - raccoglie, tra gli altri, i file per la generazione delle viste (**resources/views/**)

vendor/ - contiene, come da specifiche, tutte le dipendenze

database/ - contiene tutti i file relativi alla gestione del database, tra cui gli script per la generazione delle tabelle (**database/migrations/**), gli script per il popolamento delle tabelle (**database/seeders/**), le factory per creare massivamente contenuti di test (**database/factories/**)



Il file **.env** contiene variabili di configurazione importanti per il funzionamento dell'applicazione

Il file .env



```
APP_ENV=local
APP_KEY=base64:qgfgaLl+6zHGy/c0ifodVDS/DJ6Ew43tF0u00owh+V8=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```



Definizione delle rotte (I)

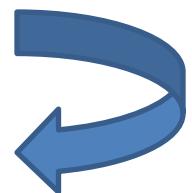


Meccanismo del *routing* – Gestione degli URL dell'applicazione Web e mapping tra URL e i metodi dei controller dell'applicazione, tramite i quali accedere alle funzionalità dell'applicazione

Meccanismo incentivato dalle politiche SEO moderne

`http://www.website.it/script.php?id=123`

`http://www.website.it/script/123`



Definizione delle rotte (II)



Le rotte sono definite nel file **routes/web.php** utilizzando il *facade Route*

- ✓ metodi corrispondenti ai metodi HTTP **get, post, put, delete, ...**
- ✓ metodi **match** o **any**
- ✓ rotte parametriche e vincoli
- ✓ gruppi di rotte

Esempi di rotte



```
Route::get('/say-hello', function(){
    return "Hello hello!";
});

Route::match(['get', 'post'], '/say-hello-multiple', function() {
    return 'This is a GET or POST request';
});

Route::any('/any', function() {
    return 'I can respond to any http method';
});

Route::get('/post/{id}', function($id) {
    return "You requested post with ID = " . $id;
});

Route::get('/optPost/{id?}', function($id = 1) {
    return "You requested post with ID = " . $id;
});

Route::group(['prefix' => 'admin'], function () {
    Route::get('users', function () {
        // Matches The "/admin/users" URL
        return "Bye bye users!";
    });

    Route::get('clients', function () {
        // Matches The "/admin/users" URL
        return "Bye bye clients!";
    });
});
```

Creazione di un controller



Classi PHP i cui metodi sono anche identificati con il termine di *azioni*

Vanno posizionati nella cartella **App\Http\Controllers** e sono sotto-classi di **Illuminate\Routing\Controller**

Sono generati tramite il comando `php artisan make:controller ${nome_controller}`

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class FrontController extends Controller
{
    public function getHome()
    {
        return view('index');
    }
}
```

Metodi HTTP e rotte RESTful



La corrispondenza tra metodi HTTP e azioni CRUD è stata nel corso degli anni standardizzata, facilitando ulteriormente la stesura di codice

- **GET** – lettura di una risorsa (se viene specificato l'ID) oppure di un insieme di risorse
- **POST** – scrittura di una risorsa (in combinazione con il metodo **GET** per la visualizzazione della form di inserimento dati)
- **PUT/PATCH** – modifica di una risorsa di cui viene specificato l'ID (in combinazione con il metodo **GET** per la visualizzazione della form di inserimento dati)
- **DELETE** – cancellazione di una risorsa (di cui viene specificato l'ID)

Routing RESTful (I)



```
Route::resource('photo', PhotoController::class);
```

Verb	Path	Action	Route Name
GET	/photo	index	photo.index
GET	/photo/create	create	photo.create
POST	/photo	store	photo.store
GET	/photo/{photo}	show	photo.show
GET	/photo/{photo}/edit	edit	photo.edit
PUT/PATCH	/photo/{photo}	update	photo.update
DELETE	/photo/{photo}	destroy	photo.destroy

```
php artisan make:controller PhotoController --resource
```

Rotte per l'esempio di riferimento



```
/*
Route::get('/', function () {
|   return view('index');
})->name('home');

*/

Route::get('/', [FrontController::class, 'getHome'])->name('home');

Route::resource('book', BookController::class);
// Placeholder for:
// - Route::get('/book', [BookController::class, 'index'])->name('book.index'); // Display the list of books
// - Route::get('/book/create', [BookController::class, 'create'])->name('book.create'); // Display the creation form
// - Route::post('/book', [BookController::class, 'store'])->name('book.store'); // Save the book in DB
// - Route::get('/book/{id}', [BookController::class, 'show'])->name('book.show'); // Display the a single book
// - Route::get('/book/{id}/edit', [BookController::class, 'edit'])->name('book.edit'); // Display the edit form
// - Route::put('/book/{id}', [BookController::class, 'update'])->name('book.update'); // Update the book in DB
// - Route::delete('/book/{id}', [BookController::class, 'destroy'])->name('book.destroy'); // Delete the book from ID

Route::resource('author', AuthorController::class);
```



Le View



- ✓ Il componente che è responsabile della visualizzazione in XHTML delle informazioni conservate nella nostra applicazione
- ✓ Laravel supporta la definizione di viste come semplici file PHP (estensione `.php`) o Blade (estensione `blade.php`), un template system molto diffuso
- ✓ Le viste sono invocate dall'interno dei metodi dei controller tramite l'helper `View`
- ✓ Le viste sono file posizionati all'interno della cartella `resources/views` e sue sottocartelle
- ✓ Per esempio, una vista nella cartella `resource/views/author/index` va referenziata con la notazione `author.index`
- ✓ Il passaggio di parametri ad una `View` viene effettuato in due modi:
 - ✓ tramite il secondo parametro (array associativo) dell'helper `View`
 - ✓ tramite il metodo `with(nome_parametro, valore_parametro)` dell'helper stesso, invocabile più volte

View e layout



- ✓ Blade, in quanto template system, permette lo sviluppo di *layout*, ovvero file con estensione **blade.php** per organizzare la struttura visuale delle pagine
- ✓ Un layout definisce un *template*, che può essere istanziato in una vista
- ✓ Nel layout, è possibile fare uso di *placeholder* attraverso la direttiva **@yield('nome_placeholder')**
- ✓ Ogni vista che istanzia un layout deve riportare all'inizio la dichiarazione **@extends('nome_layout')** (i nomi e l'organizzazione in cartelle dei layout seguono la stessa logica delle viste)
- ✓ Nella vista che istanzia un layout, un placeholder viene sostituito con codice HTML/PHP attraverso la direttiva **@section('nome_placeholder')**

```
@section('active_MyLibrary','active')

@section('breadcrumb')
<li class="breadcrumb-item" aria-current="page"><a href="{{ route('home') }}">Home</a></li>
<li class="breadcrumb-item active" aria-current="page"><a href="{{ route('book.index') }}">Library</a></li>
<li class="breadcrumb-item active" aria-current="page"><a href="{{ route('book.index') }}">Books</a></li>
<li class="breadcrumb-item active" aria-current="page">Delete book</li>
@endsection
```

Oggetto Request



L'oggetto **request** è passato come input ai metodi dei controller (azioni)

Si tratta di un'istanza della classe **Illuminate\Http\Request**

Da questo oggetto è possibile estrarre:

- Gli input dei campi di una form
- Il metodo HTTP che si sta utilizzando
- Le informazioni sugli headers del pacchetto HTTP
- I cookies scambiati tra il client e il server

Oggetto Request e form



Estrarre i valori dei campi di una form tramite l'oggetto **request**

```
$name = $request->input('name');  
$email = $request->input('email', 'mail@domain.com');  
$allParameters = $request->all();
```

Altri input dei metodi del controller sono gli eventuali parametri delle rotte parametriche

```
public function update(Request $request, $id) {  
    $dl = new DataLayer();  
    $dl->editAuthor($id, $request->input('firstName'), $request->input('lastName'));  
    return Redirect::to(route('author.index'));  
}
```

Oggetto Request e metodi HTTP



L'identificazione del metodo HTTP tramite il quale arriva la richiesta può rivelarsi utile per capire l'origine della richiesta stessa

- **GET** – form, link/bottoni, barra degli indirizzi del browser
- **POST/PUT/DELETE** – tramite form

Utile per prevenire accessi errati alle pagine e alle risorse del sito

Per estrarre il metodo HTTP tramite l'oggetto **request**

```
if ($request->isMethod('post')) {  
    ...  
}
```

Header HTTP nella richiesta



- ✓ Tramite il metodo `$request->header('')` si possono visualizzare gli headers inviati nella richiesta

```
$value = $request->header('X-Header-Name');
```

```
$value = $request->header('X-Header-Name', 'default');
```

Elenca contenuto directory

```
[Host] = localhost
[Connection] = keep-alive
[Accept] = text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
[User-Agent] = Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.63 Safari/537.31
[Referer] = http://localhost/prova/headers/
[Accept-Encoding] = gzip,deflate,sdch
[Accept-Language] = it-IT,it;q=0.8,en-US;q=0.6,en;q=0.4
[Accept-Charset] = ISO-8859-1,utf-8;q=0.7,*;q=0.3
[Cookie] = PHPSESSID=r7vuie4l3mi4t75f45001c5m23; mail=osor@osor.it
```

Oggetto Response (I)



L'oggetto **response** è la controparte dell'oggetto **request** ed è configurato all'interno dei metodi dei controller (implicitamente quando si genera una view)

Si tratta di un'istanza della classe **Illuminate\Http\Response**

```
Route::get('/', function () {
    return 'Hello World';
});
```

```
use Illuminate\Http\Response;
Route::get('/', function () {
    return new Response('Hello World');
});
```



Oggetto Response (II)



L'oggetto **response** è la controparte dell'oggetto **request** ed è configurato all'interno dei metodi dei controller (implicitamente quando si genera una view)

Si tratta di un'istanza della classe **Illuminate\Http\Response**

```
Route::get('/', function () {
    return 'Hello World';
});
```

```
Route::get('/', function () {
    return response('Hello World');
});
```



Oggetto Response (III)



Tramite l'oggetto **response** è possibile modificare gli header del pacchetto HTTP

```
return response('Hello World')
    ->header('Content-Type', 'text/plain');
```

```
return response()->view('view.name')
    ->header('Content-Type', 'text/plain');
```

Redirect (I)



Una delle azioni che è possibile compiere grazie alla modifica dell'header nella risposta HTTP:

- verso una rotta identificata tramite un nome

```
return Redirect::to(route('book.index'));
```

```
return redirect()->route('book.index');
```

```
return redirect()->route('book.edit', ['book' => $book->id]);
```

- verso la pagina precedente

```
return back()->withInput();
```

Redirect (II)



- verso un'azione specifica di un controller

```
return redirect()->action([FrontController::class, 'getHome']);
```

- verso un collegamento esterno

```
return redirect()->away('https://www.gazzetta.it/');
```

Link utili



La documentazione ufficiale di Laravel:

<https://laravel.com/docs/12.x>





PROGRAMMAZIONE WEB

**PROGRAMMAZIONE AGILE
CON LARAVEL**

Prof. Ada Bagozi

ada.bagozi@unibs.it

**P
W**



PROGRAMMAZIONE WEB

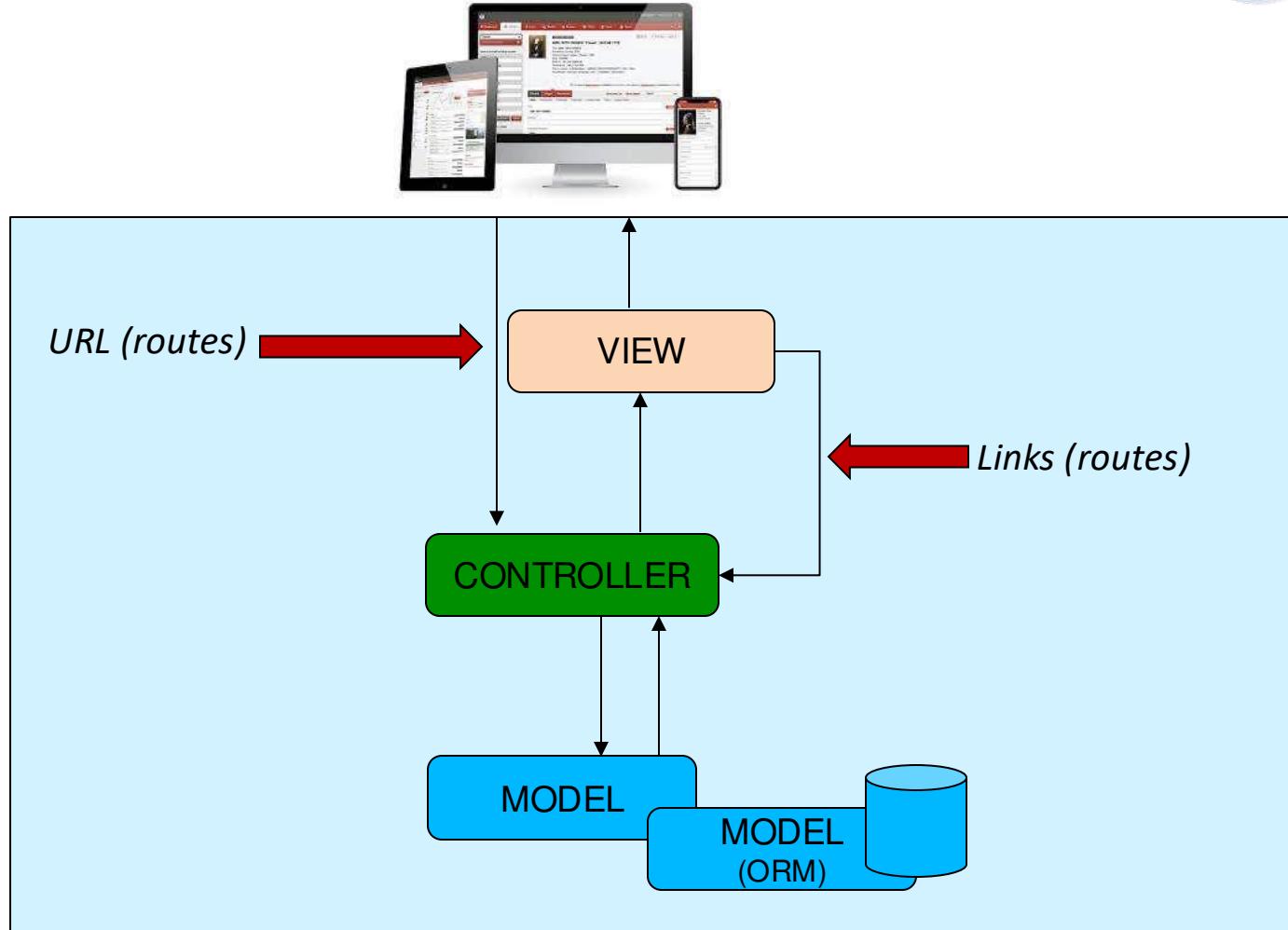
OBJECT-RELATIONAL MAPPING ELOQUENT

Prof. Ada Bagozi

ada.bagozi@unibs.it

**P
W**

MVC design pattern

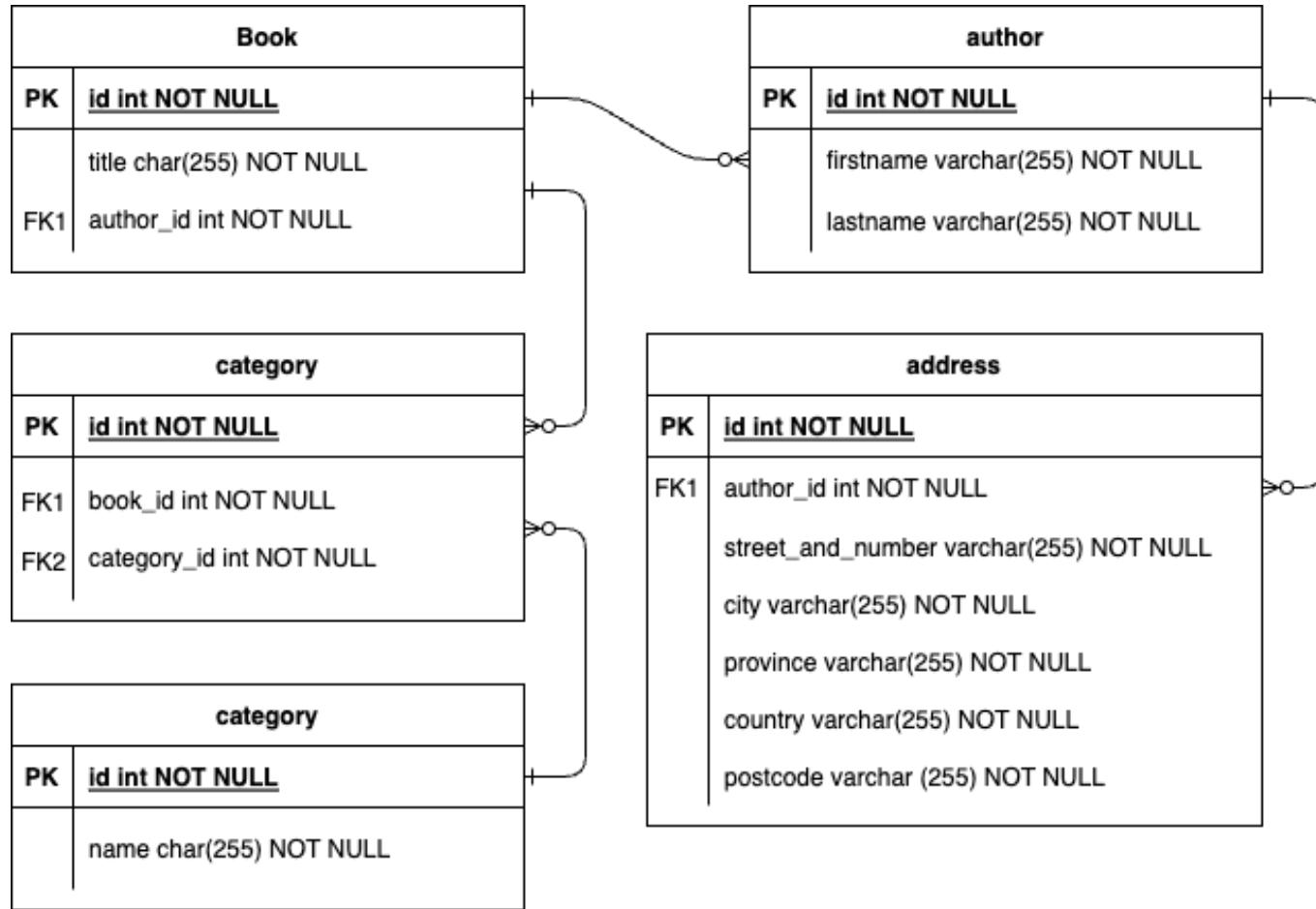


Creazione del Model (I)



- ✓ La componente **Model** include tutte le classi PHP in cui viene codificata la logica di business dell'applicazione, così come le classi per l'interazione con il database (ORM)
- ✓ Preparazione dei parametri di configurazione del database (nel file **.env**)

DB del running example



Creazione del Model (II)



```
php artisan make:model ModelName
```

Crea un file **Model.php** all'interno della cartella **app/**

Per convenzione, corrisponde ad una tabella nel database il cui nome è ottenuto da quello del modello:

- ✓ Trasformato da CamelCase a snake_case e pluralizzato (**MyAuthor** -> **my_authors**)
- ✓ A meno che non si usi la proprietà **\$table**

```
<?php  
  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
  
class Book extends Model  
{  
    //protected $table = 'alter_name_for_the_book_table';  
}
```



Creazione del Model (III)



Per convenzione, la chiave primaria:

- ✓ È associata al campo **id** autoincrementale e intero
- ✓ A meno che non si usi la proprietà **\$primaryKey**
- ✓ Ogni oggetto della classe **.php** generata presenta implicitamente una serie di campi corrispondenti alle colonne della tabella corrispondente nel DB
- ✓ Altre opzioni disponibili (**id** non incrementale, non intero, etc.)

```
<?php  
  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
  
class Book extends Model  
{  
    //protected $table = 'alter_name_for_the_book_table';  
    //protected $primaryKey = 'alter_field_as_primary_key';  
}
```



Creazione del Model (IV)



- ✓ Per default, Eloquent si aspetta in ogni tabella del database due colonne aggiuntive `created_at` e `updated_at` per tenere traccia delle firme temporali di creazione e modifica di ogni istanza
- ✓ A meno che non si usi la proprietà `public $timestamps = false`
- ✓ Altre opzioni disponibili (come la possibilità di cambiare il nome delle colonne che dovrebbero contenere le firme temporali di creazione e di modifica)
- ✓ Esiste anche la proprietà `deleted_at` per implementare il cosiddetto «soft delete»
 - ✓ Per abilitare questa opzione nel modello va esplicitato `use SoftDeletes`

Relazioni uno-a-uno



Modellata tramite i metodi **hasOne()** e **belongsTo()**

```
// Method of Author model
0 references | 0 overrides
public function address()
{
    // the property $author->address returns an object of type Address
    return $this->hasOne(Address::class, 'author_id', 'id');
}

// Method of Address model
0 references | 0 overrides
public function author()
{
    // the property $address->author returns an object of type Author
    return $this->belongsTo(Author::class, 'author_id', 'id');
}
```

La convenzione richiede una colonna **author_id** nella tabella **addresses**, in tal caso il secondo e il terzo parametro dei metodi sono evitabili

Relazioni uno-a-molti



Modellata tramite i metodi **hasMany()** e **belongsTo()**

```
// Method of Author model
0 references | 0 overrides
public function books()
{
    // the property $author->books returns an array of Books
    return $this->hasMany(Book::class,'author_id','id');
}

// Method of Book model
0 references | 0 overrides
public function author()
{
    // the property $book->author returns an object of type Author
    return $this->belongsTo(Author::class,'author_id','id');
}
```

La convenzione richiede una colonna **author_id** nella tabella **books**, che punta alla colonna id della tabella authors, in tal caso il secondo e il terzo parametro dei metodi sono evitabili

Relazioni multi-a-molti



Modellata tramite il metodo **belongsToMany()**

```
// Method of Book model
5 references | 0 overrides
public function categories()
{
    // the property $book->categories returns an array of Category
    return $this->belongsToMany(Category::class, 'book_category', 'book_id', 'category_id');
}

// Method of Category model
0 references | 0 overrides
public function books()
{
    // the property $category->books returns an array of Book
    return $this->belongsToMany(Book::class, 'book_category', 'category_id', 'book_id');
}
```

La convenzione richiede una tabella **book_category** con le due colonne **book_id** e **category_id**; in tal caso, i parametri aggiuntivi possono essere omessi

Creazione delle tabelle nel DB (I)



Laravel offre il meccanismo delle *migration* – Si tratta di script che sono posizionati nella cartella **database/migrations/** e si occupano del versioning del database tramite i metodi **up()** e **down()**

```
php artisan make:migration book_table
```

A questo punto, il comando `php artisan migrate` genererà le tabelle nel DB
Per annullare le migration eseguite sull'applicazione il comando da utilizzare è `php artisan migrate:rollback`

Creazione delle tabelle nel DB (II)



```
Schema::create('author', function (Blueprint $table) {
    $table->id();
    $table->string('firstname');
    $table->string('lastname');
    $table->timestamps();
});
```

```
Schema::create('book', function (Blueprint $table) {
    $table->id();
    $table->string('title');
    $table->unsignedBigInteger('author_id');
    $table->timestamps();
});
```

```
Schema::table('book', function (Blueprint $table) {
    $table->foreign('author_id')->references('id')->on('author');
});
```



Popolamento delle tabelle nel DB



Laravel offre anche gli strumenti (denominati *Factory* e *Seed*) per popolare le tabelle del DB con dati di test

- Le factory sono classi che si occupano di costruire istanze di particolari modelli inserendo dati di test nelle loro proprietà (cartella **database/factories**)
- I seed sono script per il popolamento del database (che possono far uso delle factory) attraverso il comando `php artisan make:seeder ${nome_seeder}`, che genererà uno script `${nome_seeder}` nella cartella **database/seeds**; il seeder viene invocato con il comando `php artisan db:seed --class=${nome_seeder}`

Seeder e factories possono essere create anche contemporaneamente al modello

Factories - Esempio



```
php artisan make:model Model --factory
```

```
namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;
use App\Models\Book;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Book>
 */
0 references | 0 implementations
class BookFactory extends Factory
{
    0 references
    protected $model = Book::class;
    0 references | 0 overrides
    public function definition(): array
    {
        return [
            'title' => $this->faker->sentence(rand(1, 5))
        ];
    }
}
```



Seeders - Esempio (I)



```
class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    0 references | 0 overrides
    public function run(): void
    {
        $this->populateDB();
        $this->createUsers();
    }

    1 reference
    private function populateDB()
    {

        // Create 100 authors with their corresponding address
        Author::factory()->count(100)->create()->each(function ($author) {
            Address::factory()->count(1)->create(['author_id' => $author->id]);
        });

        // Randomly select a subset of 50 authors and, for each of them, create a set of books (from 1 to 5, randomly generated)
        $authors = Author::all();
        $authorsWithBooks = $authors->random(50);

        foreach($authorsWithBooks as $singleAuthor) {
            $numberOfBooks = rand(1,5);
            for($b=0; $b<$numberOfBooks; $b++) {
                Book::factory()->count(1)->create(['author_id' => $singleAuthor->id]);
            }
        }
    }
}
```



Seeders - Esempio (II)



```
// Create 10 book categories
Category::factory()->count(1)->create(['name' => 'Romanzi classici']);
Category::factory()->count(1)->create(['name' => 'Fantasy']);
Category::factory()->count(1)->create(['name' => 'Gialli']);
Category::factory()->count(1)->create(['name' => 'Thriller']);
Category::factory()->count(1)->create(['name' => 'Saggi']);
Category::factory()->count(1)->create(['name' => 'Poesie']);
Category::factory()->count(1)->create(['name' => 'Psicologia']);
Category::factory()->count(1)->create(['name' => 'Fantascienza']);
Category::factory()->count(1)->create(['name' => 'Viaggi']);
Category::factory()->count(1)->create(['name' => 'Arte e fotografia']);

// Randomly associate a book to a subset of categories (from 1 to 5)
$books = Book::all();
$categories = Category::all();

foreach($books as $singleBook)
{
    $numberOfCategories = rand(1,5);
    $selectedCategories = $categories->random($numberOfCategories);
    $singleBook->categories()->attach($selectedCategories);
}
```



Per l'inserimento massivo...



Proprietà modificabili in modalità *massiva* (o non modificabili):

```
class Book extends Model
{
    protected $table = 'book';
    //protected $primaryKey = 'alter_field_as_primary_key';
    //use SoftDeletes;
    public $timestamps = false;
    use HasFactory;

    protected $fillable = ['title', 'author_id'];
```

Eloquent – Azioni CRUD (I)



```
class DataLayer
{
    1 reference | 0 overrides
    public function listBooks() {
        $books = Book::orderBy('title','asc')->get();
        return $books;
    }

    3 references | 0 overrides
    public function findBookById($id) {
        return Book::find($id);
    }

    2 references | 0 overrides
    public function listAuthors() {
        $authors = Author::orderBy('lastname','asc')->orderBy('firstname','asc')->get();
        return $authors;
    }

    0 references | 0 overrides
    public function findAuthorById($id) {
        return Author::find($id);
    }

    2 references | 0 overrides
    public function getAllCategories() {
        return Category::orderBy('name','asc')->get();
    }
}
```



Eloquent – Azioni CRUD (II)



```
1 reference | 0 overrides
public function addBook($title, $author_id, $categories) {
    $book = new Book;
    $book->title = $title;
    $book->author_id = $author_id;
    $book->save();
    foreach($categories as $cat) {
        $book->categories()->attach($cat);
    }
    // massive creation (only with fillable property enabled on Book):
    // Book::create(['title' => $title, 'author_id' => $author_id, 'user_id' => $user]);
}

0 references | 0 overrides
public function addAuthor($first_name, $last_name) {
    $author = new Author;
    $author->firstname = $first_name;
    $author->lastname = $last_name;
    $author->save();

    //use the factory to randomly generate an address
    Address::factory()->count(1)->create(['author_id' => $author->id]);

    // massive creation (only with fillable property enabled on Author):
    // Author::create(['firstname' => $first_name, 'lastname' => $last_name, 'user_id' => $user]);
}
```



Eloquent – Azioni CRUD (III)



```
1 reference | 0 overrides
public function editBook($id, $title, $author_id, $categories) {
    $book = Book::find($id);
    $book->title = $title;
    $book->author_id = $author_id;
    $book->save();

    // Cancel the previous list of categories
    $prevCategories = $book->categories;
    foreach($prevCategories as $prevCat) {
        $book->categories()->detach($prevCat->id);
    }

    // Update the list of categories
    foreach($categories as $cat) {
        $book->categories()->attach($cat);
    }
    // massive update (only with fillable property enabled on Book):
    // Book::find($id)->update(['title' => $title, 'author_id' => $author_id]);
}

0 references | 0 overrides
public function editAuthor($id, $first_name, $last_name) {
    $author = Author::find($id);
    $author->firstname = $first_name;
    $author->lastname = $last_name;
    $author->save();
    // massive update (only with fillable property enabled on Author):
    // Author::find($id)->update(['firstname' => $first_name, 'lastname' => $last_name]);
}
```

Eloquent – Azioni CRUD (IV)



```
1 reference | 0 overrides
public function deleteBook($id) {
    $book = Book::find($id);
    $categories = $book->categories;
    foreach($categories as $cat) {
        $book->categories()->detach($cat->id);
    }
    $book->delete();
}
```

```
0 references | 0 overrides
public function deleteAuthor($id) {
    $author = Author::find($id);
    $author->address->delete();
    $author->delete();
}
```

Running example v5 (I)



Screenshot of a web application interface titled "Bibrios :: Books' List". The top navigation bar includes a logo, "Home", and "My Library" dropdown. Below the navigation is a breadcrumb trail: "Home / Library / Books".

Title	Author	Actions
Ab itaque quod qui.	Adams	Details Edit Delete
Aliquid adipisci dolor.	Boyer	Details Edit Delete
Amet laboriosam temporibus enim.	Rutherford	Details Edit Delete

[Create new book](#)

```
Route::get('/book', [BookController::class, 'index'])->name('book.index'); // Display the list of books
```



Running example v5 (II)



Home My Library ▾

Home / Library / Books / Ab itaque quod qui.

Biblios :: book details

Title:	Ab itaque quod qui.	<input checked="" type="button"/> Edit
Author:	Adams, Fritz	<input type="button"/> Delete
Categories:	Arte e fotografia Gialli Thriller Viaggi	

Back

```
Route::get('/book/{id}', [BookController::class, 'show'])->name('book.show'); // Display a single book
```

PW

Running example v5 (III)



Screenshot of a web application interface:

- Header: Home | My Library ▾
- Breadcrumbs: Home / Library / Books / Edit book
- Title: Biblios :: Edit Book
- Form fields:
 - Categories: Arte e fotografia, Fantascienza, Fantasy, Gialli
 - Title: Ab itaque quod qui.
 - Author: Adams
- Action button: Save

```
Route::get('/book/{id}/edit', [BookController::class, 'edit'])->name('book.edit'); // Display the edit form

@if(isset($book->id))
    <form class="form-horizontal" name="book" method="post" action="{{ route('book.update', ['book' => $book->id]) }}">
        <!--<input type="hidden" name="_method" value="PUT">-->
        @method('PUT')
@else
    <form class="form-horizontal" name="book" method="post" action="{{ route('book.store') }}">
@endif
@csrf
```



Running example v5 (IV)



Deleting book "Ab itaque quod qui." from the list

Deleting book. Confirm?

<p>Revert</p> <p>The book will not be removed from the data base</p> <p> Cancel</p>	<p>Confirm</p> <p>The book will be permanently removed from the data base</p> <p> Delete</p>
---	--

```
Route::delete('/book/{id}', [BookController::class, 'destroy'])->name('book.destroy'); // Delete the book from ID
Route::get('/book/{id}/destroy/confirm', [BookController::class, 'confirmDestroy'])->name('book.destroy.confirm');
```

```
<form name="book" method="post" action="{{ route('book.destroy', ['book' => $book->id]) }}>
    @method('DELETE')
    @csrf
    <label for="mySubmit" class="btn btn-danger"><i class="bi bi-trash"></i> Delete</label>
    <input id="mySubmit" class="d-none" type="submit" value="Delete">
</form>
```

Link utili



La documentazione ufficiale di Eloquent inclusa in quella di Laravel: <https://laravel.com/docs/12.x/eloquent>





PROGRAMMAZIONE WEB

OBJECT-RELATIONAL MAPPING ELOQUENT

Prof. Ada Bagozi

ada.bagozi@unibs.it





PROGRAMMAZIONE WEB

SCRIPTING CLIENT-SIDE IN JAVASCRIPT INTRODUZIONE

Prof. Ada Bagozi

ada.bagozi@unibs.it

PW



LE BASI DEL LINGUAGGIO



Javascript - Definizione



- ✓ **JavaScript** è un linguaggio di scripting lato-client, ossia un linguaggio di programmazione interpretato dal browser che prevede la scrittura di **script**
- ✓ Uno **script** è un piccolo programma (contenuto o importato in una pagina HTML) che viene **interpretato** ed **eseguito** dal browser
- ✓ Mediante l'uso di script è possibile **creare dinamicamente** i contenuti di una pagina web e aggiungere **interattività** alla pagina stessa (con un occhio alle prestazioni)

- ✓ **Javascript NON È Java:** JavaScript e Java sono due linguaggi di programmazione differenti!!!

Javascript – Lo script



Gli script JavaScript possono essere:

- ✓ contenuti in uno o più file di testo con estensione **js** e linkati al file HTML con il tag **script** che va inserito fra i tag **head**

```
<head>
  <script type="text/javascript"
         src="myScript.js"/>
```

```
<head>
```

- ✓ contenuti nel file HTML, inseriti come testo all'interno del tag script

```
<script type="text/javascript"> ... </script>
```



Inserire lo script in una pagina



- ✓ Esistono inoltre alcuni attributi HTML in cui si può incorporare del codice:
 - ✓ gli attributi per la gestione degli eventi, come **onclick**, possono contenere frammenti di codice (ma non dichiarazioni), da eseguire al verificarsi dell'evento
 - ✓ L'attributo **href** del tag **<a>** può fare riferimento a una funzione javascript con la sintassi:
javascript:nome_funzione(parametri);
in questo caso, il click del link eseguirà la chiamata alla funzione

Esecuzione di uno script



- ✓ Tutte le funzioni e le variabili dichiarate negli script diventano disponibili (quindi possono essere usate e chiamate) non appena il parser analizza il punto della pagina in cui sono dichiarate
- ✓ Se uno script contiene codice immediato, cioè scritto al di fuori di funzioni, questo viene eseguito non appena il parser analizza il punto della pagina in cui il codice compare
- ✓ Gli script possono utilizzare liberamente funzioni e variabili dichiarate in altri script inseriti nella stessa pagina.

Partiamo dalla sintassi ...



JavaScript è un linguaggio di programmazione **case sensitive** ossia fa distinzione tra lettere maiuscole e minuscole

num è diverso da **Num**

Ogni singola istruzione va conclusa con il punto e virgola

alert("Hello world!!!");

I commenti all'interno di uno script vanno inseriti tra i caratteri **/*** e ***/** oppure dopo **//** per commenti su una riga

/* questo è un commento */

Tipi di dato



In Javascript possiamo avere i seguenti tipi di dati

Numeri - in JavaScript non vi è differenza tra numeri interi e numeri in virgola mobile

Stringhe - una stringa è formata da una sequenza di zero o più caratteri racchiusi tra apici singoli o doppi (‘ o “):

“casa” è la stringa casa

“casa ‘Pisa’” è la stringa casa ‘Pisa’

‘casa “Pisa”’ è la stringa ‘casa “Pisa”’

Valori Booleani - è un dato che esprime un “valore di verità” e può assumere solo due valori **true** e **false**

Array o Oggetti

Dichiarazione di variabili



Una variabile viene dichiarata mediante l'uso della parola chiave **var**:

var i; dichiara la variabile **i**

var j = 0;

dichiara la variabile **j** e le assegna il valore **0**

var s = "casa";

dichiara la variabile **s** e le assegna il valore **"casa"**

Se una variabile viene ri-dichiarata, non perde il suo valore

Variabili locali e globali



`var s = "pluto";` variabile **s** locale di tipo stringa con valore iniziale
"pluto"

`var n = 3;` variabile **n** locale di tipo numerico con valore 3

`t = "paperino";` variabile **t** globale di tipo stringa con valore iniziale
"paperino"

`m = n;` variabile **m** globale di tipo numerico con valore 3

`u = v;` la variabile **u** ha valore undefined (in quanto **v** non è a sua volta definita)

`var b = (3>2);` variabile **b** locale booleana con valore **true**

`var o = new Object();` variabile **o** locale di tipo **Object**
(vuota)

Funzioni in Javascript (I)



Una **funzione** racchiude una porzione di codice JavaScript che può essere eseguito e viene definita mediante la parola chiave **function** nel seguente modo:

```
function nomeFunzione (par1, par2)
{
    istruzioni;
    ...
}
```

Funzioni in Javascript (II)



- ✓ Le funzioni Javascript sono in realtà variabili con valore di tipo **Function**
- ✓ Per fare riferimento a una funzione è sufficiente usare il suo nome, o un'espressione equivalente che abbia valore di tipo **Function**
- ✓ Una volta ottenuto il riferimento a una funzione è possibile:
 - ✓ chiamare la funzione passandole una lista di parametri
 - ✓ omettere uno o più parametri al termine della lista; in questo caso, tali parametri varranno **undefined** nel corpo della funzione
 - ✓ passare come argomento una funzione ad un'altra funzione
 - ✓ assegnare una funzione a una o più variabili
 - ✓ accedere a tutti gli elementi della funzione, per modificarla o ridefinirla, tramite le proprietà di **Function**
 - ✓ verificare se una funzione è definita come si farebbe con qualsiasi variabile, ad esempio testandola con un **if (nome_funzione)**

Funzioni in Javascript (III)



- ✓ Le funzioni restituiscono il controllo al chiamante al termine del loro blocco di istruzioni
- ✓ È possibile restituire un valore al chiamante, in modo da poter usare la funzione in espressioni più complesse, utilizzando la sintassi **return espressione**
- ✓ L'espressione può essere di qualsiasi tipo; essa viene valutata e il valore risultante è restituito
- ✓ Se la funzione non restituisce nessun valore, Javascript sottintende un "**return undefined**"隐式

Funzioni - Esempi



//funzione con due parametri, dichiarazione diretta

```
function prodotto(a,b) {  
    return a*b;  
}
```

//oggetto funzione assegnato a una variabile

```
var per = new Function("a","b","return a*b;");
```

```
<script type="text/javascript">  
function prodotto(a,b)  
{  
    return a*b;  
}  
</script>  
</head>  
  
<body>  
<script type="text/javascript">  
var per = new Function("a","b","return a*b;");  
document.write(per(4,3));  
document.write("<br>");  
document.write(prodotto(4,3));  
</script>
```

Funzioni – Passaggio di parametri



- ✓ Il passaggio dei parametri alle funzioni Javascript avviene in maniera diversa a seconda del tipo del parametro stesso:
 - ✓ i tipi booleano, stringa, numero e undefined sono passati per valore, cioè nella funzione è presente una copia del valore usato come argomento; cambiamenti locali alla funzione non influenzano il valore dell'argomento usato nella chiamata alla funzione stessa
 - ✓ il tipo oggetto è passato per riferimento; la manipolazione del contenuto dell'oggetto si riflette sull'oggetto usato come argomento

Javascript - Oggetti



- ✓ In Javascript non si possono definire classi, ma solo speciali funzioni dette costruttori che creano oggetti con proprietà e metodi; il nome della funzione costruttore è considerato il nome della classe dell'oggetto
- ✓ Gli oggetti si creano utilizzando l'operatore **new** applicato alla loro funzione costruttore: **`o = new Object()`**
- ✓ Un metodo di creazione alternativo consiste nell'utilizzo del costrutto **`{"proprietà": "valore", ...}`**, che crea un oggetto con le proprietà date

Proprietà degli oggetti



- ✓ Le proprietà di un oggetto Javascript possono contenere valori di qualsiasi tipo
- ✓ Per accedere a una proprietà, si possono usare due sintassi:
 - ✓ Sintassi “a oggetti”: **oggetto.proprietà**
 - ✓ Sintassi “array”: **oggetto[“proprietà”]**
- ✓ Se si tenta di leggere il valore di una proprietà non definita in un oggetto, si ottiene il valore **undefined** (come per ogni variabile non assegnata)
- ✓ È possibile aggiungere dinamicamente proprietà agli oggetti semplicemente assegnando loro un valore
- ✓ Non è possibile aggiungere proprietà a variabili che non siano di tipo oggetto (cioè che non siano oggetti predefiniti o creati con **new**)

Proprietà degli oggetti - Esempi



```
var o = new Object(); // creazione esplicita di un oggetto
var v = o.pippo; // v è undefined
o.pluto = 3; // adesso o ha una proprietà pluto, di tipo Number,
              con valore 3
v = o.pluto; // adesso v è una variabile di tipo Number e vale 3
v.paperino = "ciao"; // è un errore: una variabile può accettare
                      l'aggiunta di proprietà solo se è di tipo oggetto
var o2 = {"pippo": "ciao", "pluto": 3}; // creazione
                                         implicita di un oggetto
v = o2["pluto"]; // equivalente a v = o2.pluto
var nome = "pippo"; // adesso nome è una variabile di tipo
                     String e vale "pippo"
v=o2[nome]; // accesso a una proprietà con nome dinamico
             assegnato a una seconda variabile
```

Oggetti Javascript - Metodi



- ✓ I metodi di un oggetto Javascript sono semplicemente proprietà di tipo **function**
- ✓ Per accedere a un metodo si possono usare le stesse sintassi viste per le proprietà
- ✓ Per aggiungere un metodo a un oggetto, è sufficiente creare una proprietà col nome del metodo ed assegnarvi
 - una funzione già definita
 - una funzione anonima: **function (parametri) {corpo}**
- ✓ I metodi possono essere aggiunti in qualsiasi momento a un oggetto, esattamente come le proprietà
- ✓ I metodi di un oggetto, per far riferimento alle proprietà dell'oggetto in cui sono definiti, devono utilizzare la parola chiave **this**:
this.proprietà

Oggetti e Metodi - Esempi



```
var o = new Object();

o.metodo1 = function(x) {return x;} // aggiunge la funzione
// specificata come metodo1 all'oggetto

o["metodo2"] = f; // aggiunge la funzione f (se esistente) come metodo2
// all'oggetto

o.metodo1 // questa espressione restituisce l'oggetto function che
// rappresenta il metodo1

o.metodo1(3);

o["metodo1"](3); // due chiamate equivalenti al metodo1

var o2 = {"pippo": "ciao", "pluto": 3, "metodo3":
    function(x) {return x;}} // definizione di un metodo all'interno della
// sintassi di creazione

var o3 = new Object();

o3.metodo3 = o.metodo1 // il metodo3 dell'oggetto o3 è una copia del
// metodo1 dell'oggetto o
```

Altri esempi



```
function myObject(a) {  
    this.v = a+1;  
    this.w = 0;  
    this.m = function(x) {return this.v+x;}  
}
```

```
var o = new myObject(2);  
/*
```

L'oggetto **o** avrà due proprietà (**v** e **w**), una delle quali inizializzata tramite il parametro della funzione costruttore, e un metodo (**m**) che restituisce il valore della proprietà **v** sommata al suo argomento

```
*/  
o.m(3); // ritorna 6  
o.getW = function() {return this.w;} // aggiunta dinamica di membri all'oggetto  
(NON al costruttore)  
o.getV = function() {return v;} // SBAGLIATO! Fa riferimento alla variabile  
GLOBALE v!
```



Gestione delle eccezioni



- ✓ Nelle versioni più recenti di Javascript è stato introdotto anche un sistema di gestione delle eccezioni in stile Java
- ✓ Un'eccezione segnala un imprevisto, spesso un errore, all'interno della normale esecuzione del codice
- ✓ Un'eccezione può venire sollevata dalle librerie di Javascript o dal codice scritto dall'utente, attraverso la parola chiave **throw**
- ✓ Per gestire le eccezioni, è possibile avvalersi del costrutto **try...catch...finally**

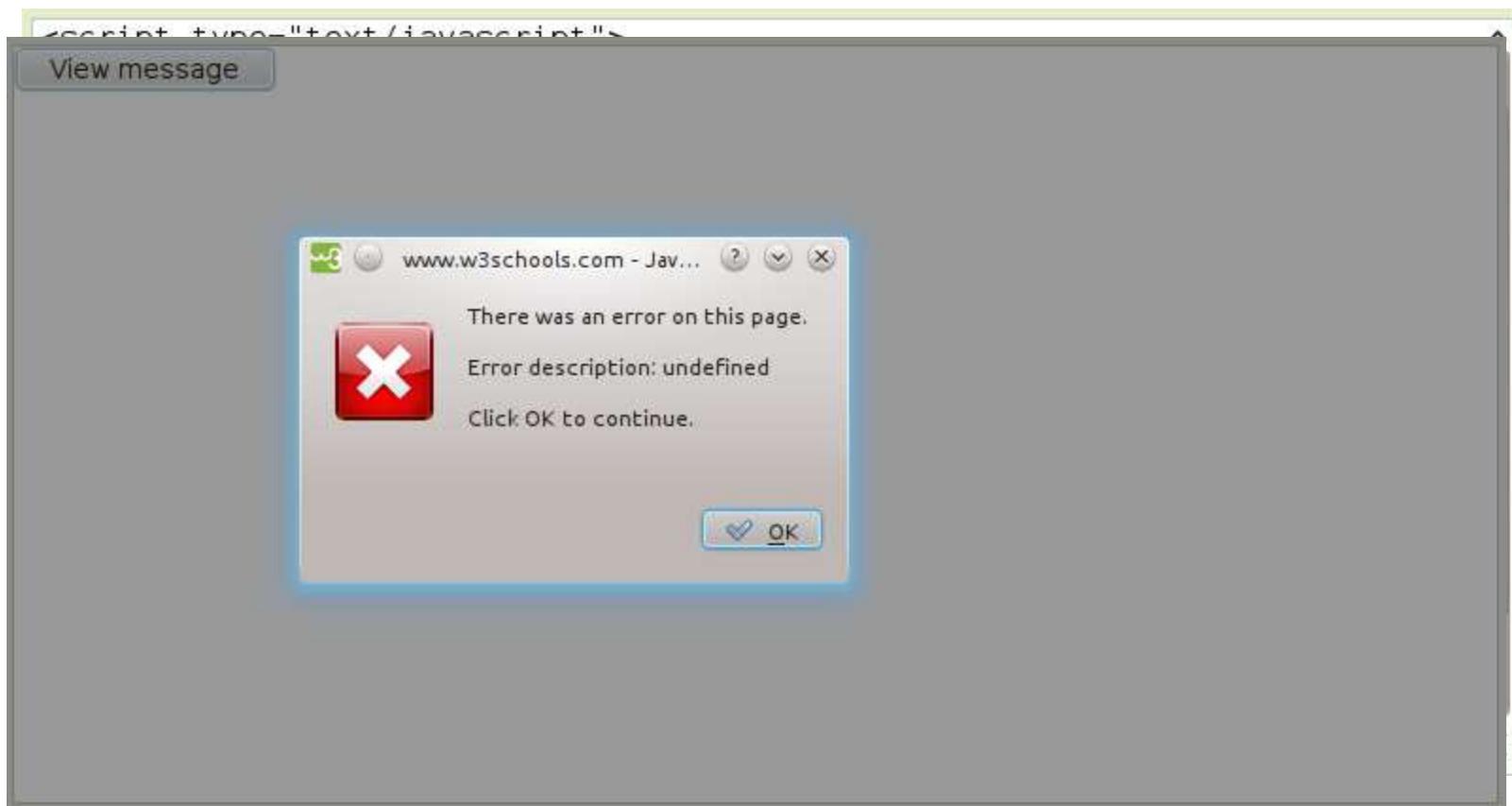
Gestione delle eccezioni - Esempio



```
<script type="text/javascript">
var txt="";
function message()
{
try
{
  adddler("Welcome guest!");
}
catch(err)
{
  txt="There was an error on this page.\n\n";
  txt+="Error description: " + err.description + "\n\n";
  txt+="Click OK to continue.\n\n";
  alert(txt);
}
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
```



Gestione delle eccezioni - Esempio



La clausola throw

```
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","");
try
{
  if(x>10)
  {
    throw "Err1";
  }
  else if(x<0)
  {
    throw "Err2";
  }
  else if(isNaN(x))
  {
    throw "Err3";
  }
}
catch(er)
{
  if(er=="Err1")
  {
    alert("Error! The value is too high");
  }
  if(er=="Err2")
  {
    alert("Error! The value is too low");
  }
  if(er=="Err3")
  {
    alert("Error! The value is not a number");
  }
}
</script>
</body>
</html>
```



Perché Javascript?



- ✓ JavaScript è utilizzato per:
 - ✓ verificare la *correttezza e la completezza dell'input dell'utente* (tipicamente attraverso controlli sui valori inseriti nei campi di una form)
 - ✓ implementare *animazioni* all'interno delle pagine HTML (modificando “on-the-fly” il codice HTML della pagina)
 - ✓ intercettare e gestire le interazioni degli utenti con le pagine HTML, attraverso un *modello “ad eventi”*
 - ✓ minimizzare la frequenza di interazioni tra client e server, gestendo lato client le operazioni che non è necessario veicolare lato server
- ✓ A tal fine, Javascript si “appoggia” sul modello del *DOM (Document Object Model)* e sull’uso della *tecnologia AJAX*
- ✓ Utilizzeremo le potenzialità di Javascript attraverso l’utilizzo di jQuery, una libreria molto diffusa



DOCUMENT OBJECT MODEL

L'ambiente del Web browser



Esistono due modelli ad oggetti per definire l'interfaccia e i vari aspetti del browser e del documento manipolabili con Javascript

- ✓ un modello ad oggetti del browser (**BOM – Browser Object Model**)
- ✓ un modello ad oggetti del documento (**DOM – Document Object Model**)

Esiste un modello di programmazione azionato dagli eventi (*event-driven*)

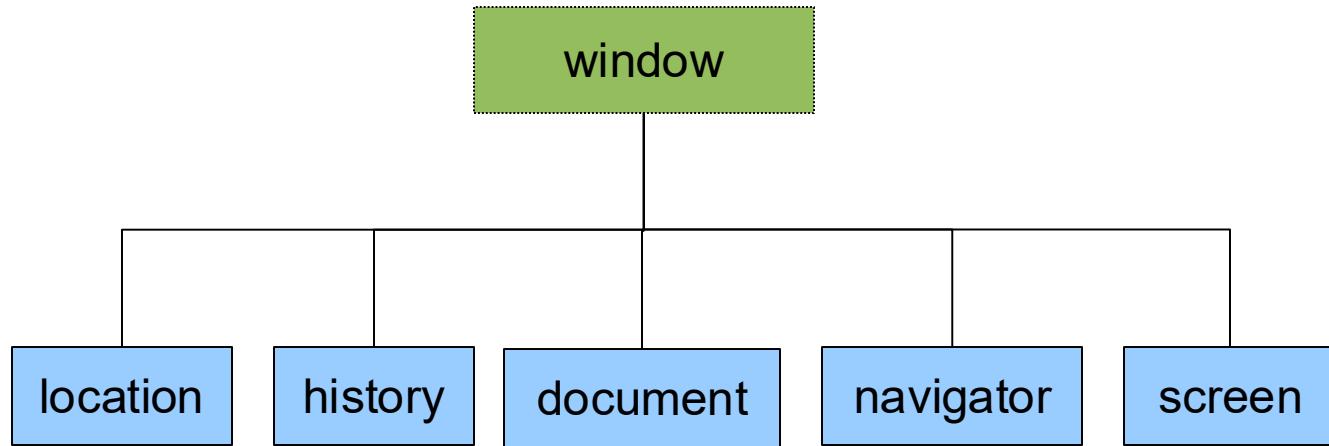
- ✓ ad esempio, se si preme un bottone o si carica una pagina si genera un evento che può essere controllato (tramite opportuni *event handlers*)

Il BOM – Browser Object Model



- ✓ Permette di interagire con il browser, offrendo l'accesso a varie caratteristiche dell'ambiente in cui il browser è in esecuzione: la finestra del browser, le caratteristiche dello schermo, la cronologia del browser, etc.
- ✓ Nel modello BOM va posta molta attenzione alle problematiche *cross-browser*
- ✓ L'oggetto di base per ottenere tutto questo è l'oggetto **window**
- ✓ Tra i metodi più comuni dell'oggetto **window** troviamo **alert()**, **prompt()**, **confirm()** (si vedano gli approfondimenti)

L'albero del BOM



Per maggiori info: https://www.w3schools.com/js/js_window.asp

L'albero del DOM



- ✓ Possiamo rappresentare ogni documento HTML attraverso un albero
- ✓ Alla radice c'è il nodo che rappresenta il tag **HTML**, i suoi figli sono i nodi che rappresentano **HEAD** e **BODY**
- ✓ DOM Livello 1 fornisce le API per accedere ai nodi dell'albero

Esempio - HTML

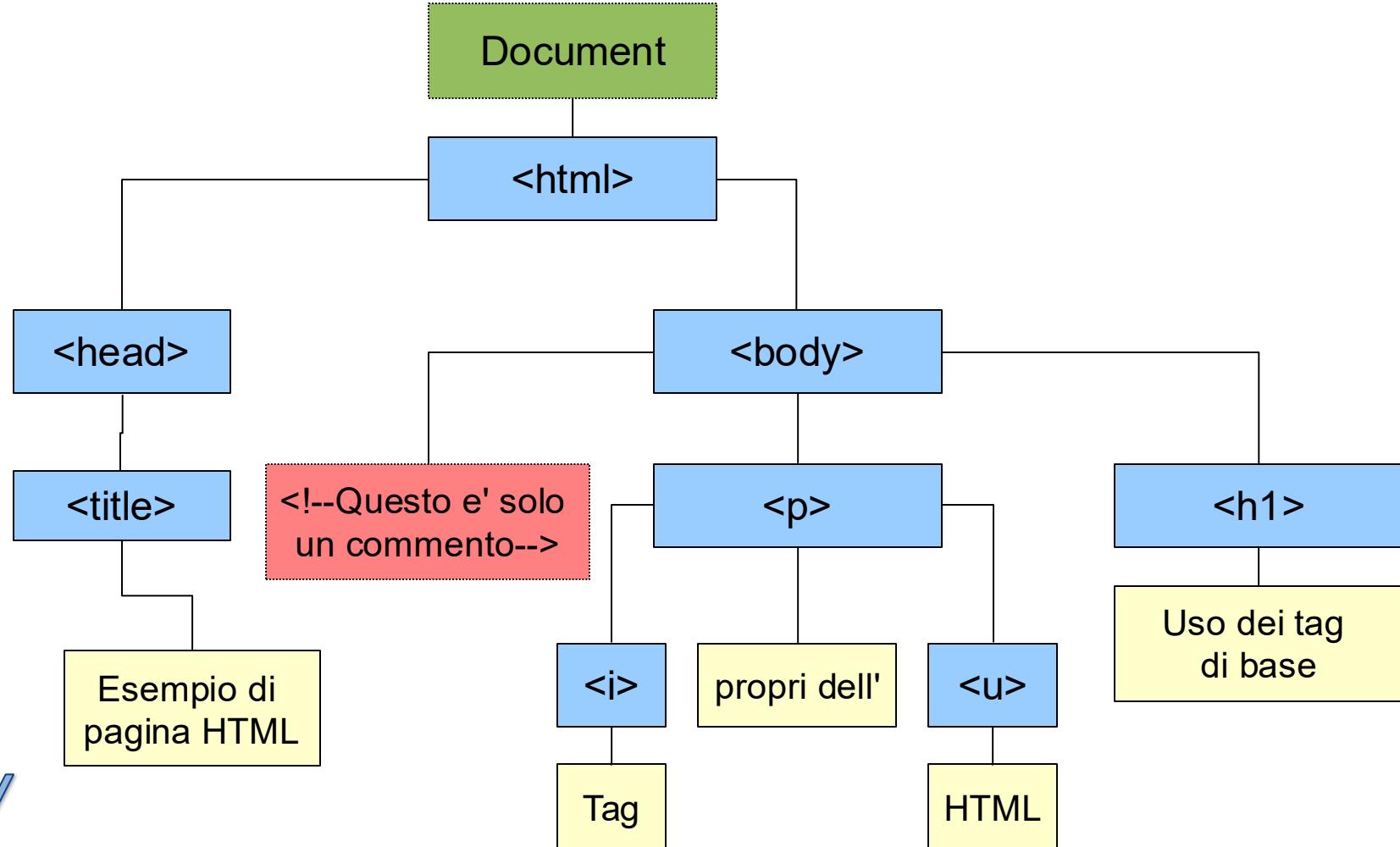


```
<html>
  <head>
    <title>Esempio di pagina HTML</title>
  </head>

  <body>
    <!-- questo e' solo un commento -->
    <h1>Uso dei tag di base</h1>
    <p><i>Tag</i> propri dell'<u>HTML</u></p>
  </body>

</html>
```

HTML visto come albero



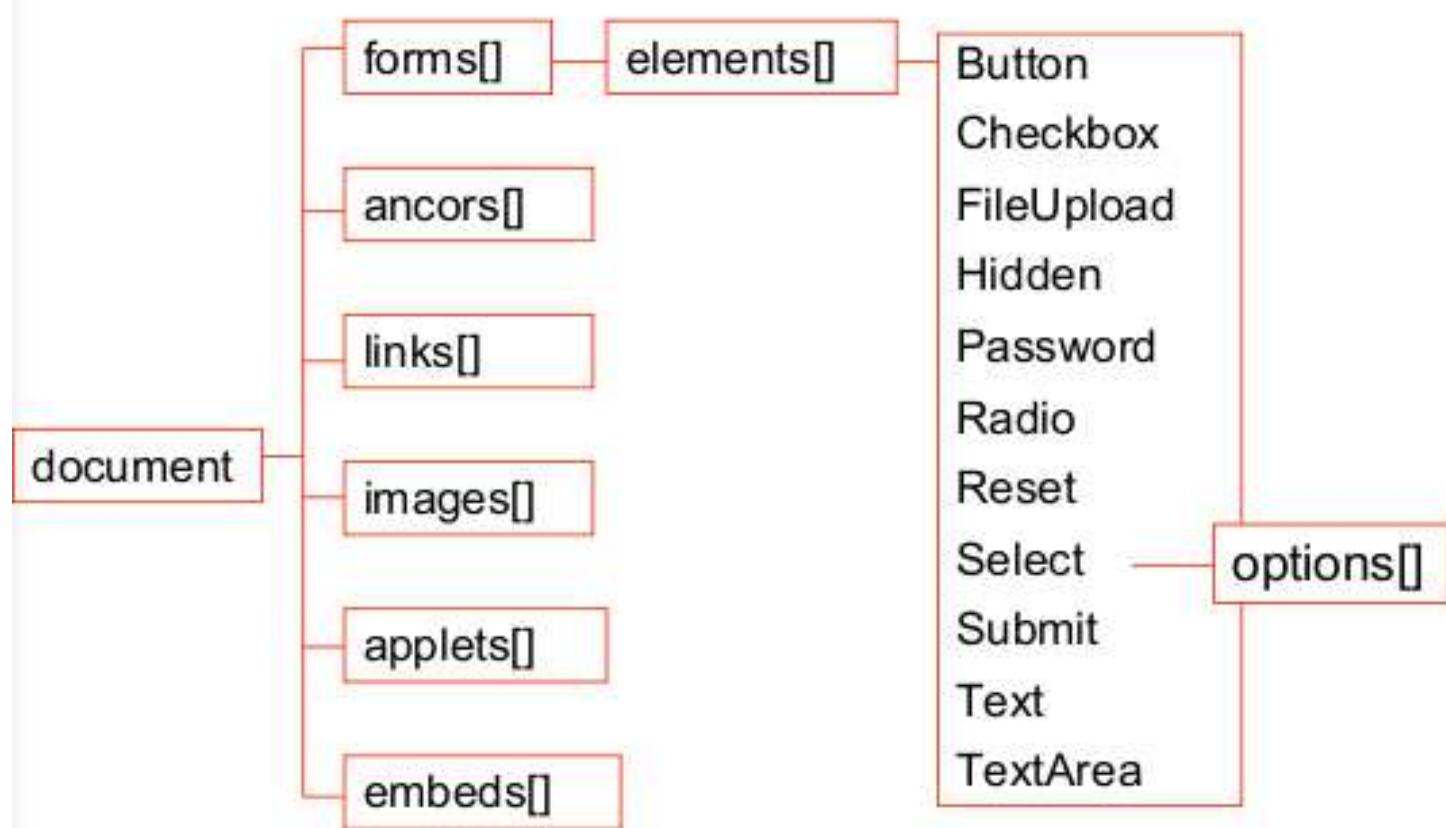
Il DOM – Document Object Model



Fornisce l'accesso al **contenuto** della finestra del browser, all'interno dei tag HTML

- ✓ **DOM Level 0** – spesso chiamato modello ad oggetti **tradizionale** o **classico**, permette l'accesso ad alcuni elementi particolari, come immagini, link, elementi di una form, etc.
- ✓ **DOM Level 1** – permette di accedere a tutti gli elementi HTML, manipolando i **nodi** del documento tramite l'oggetto **document**
- ✓ **DOM Level 2** – permette di far riferimento alle proprietà CSS degli elementi, per esempio per implementare animazioni

DOM – Level 0



Accedere agli elementi del DOM



Per accedere agli elementi del DOM si utilizza la classica dot notation

Ad esempio, con l'espressione seguente

- ✓ `document.forms[0].elements[3].options[2].text`

- ✓ accediamo al testo (`text`) del terzo item di una lista di selezione (`options[2]`) che è il quarto elemento (`elements[3]`) della prima form (`forms[0]`) del documento

Il ciclo `for..in`



Il ciclo `for..in` è utilizzato per processare in maniera ciclica le proprietà di un oggetto

```
for (variable in object)
{
    code to be executed
}
```

Il ciclo for..in - Esempio



```
<html>
  <head>
    <title>Esempi Javascript – Window properties</title>
  </head>
  <body>
    <script type="text/javascript">
      var prop = "";
      var i = 0;
      for (prop in window) {
        i++;
        document.write(i + ". <b>");
        document.write(prop + "</b>");
        document.write(window[prop] + "<br>");
      }
    </script>
  </body>
</html>
```

Il ciclo for..in - Esempio



```
1. window:[object Window]
2. self:[object Window]
3. document:[object HTMLDocument]
4. name:
5. location:https://elearning.unibs.it/pluginfile.php/836621/mod_resource/content/19/windowProperties.html
6. customElements:[object CustomElementRegistry]
7. history:[object History]
8. navigation:[object Navigation]
9. locationbar:[object BarProp]
10. menubar:[object BarProp]
11. personalbar:[object BarProp]
12. scrollbars:[object BarProp]
13. statusbar:[object BarProp]
14. toolbar:[object BarProp]
15. status:
16. closed:false
17. frames:[object Window]
18. length:0
19. top:[object Window]
20. opener:null
21. parent:[object Window]
22. frameElement:null
23. navigator:[object Navigator]

229. i:229
230. TEMPORARY:0
231. PERSISTENT:1
232. addEventListener:function addEventListener() { [native code] }
233. dispatchEvent:function dispatchEvent() { [native code] }
234. removeEventListener:function removeEventListener() { [native code] }
```



PROGRAMMAZIONE WEB

SCRIPTING CLIENT-SIDE IN JAVASCRIPT INTRODUZIONE

Prof. Ada Bagozi

ada.bagozi@unibs.it





PROGRAMMAZIONE WEB

SCRIPTING CLIENT-SIDE IN JAVASCRIPT LA LIBRERIA JQUERY

Prof. Ada Bagozi

ada.bagozi@unibs.it



Cos'è JQuery?



JQuery è una libreria Javascript per la programmazione *crossbrowser avanzata* «costruita» attorno al DOM

JQuery si affianca a molte librerie simili, ma presenta caratteristiche notevoli:

- ✓ è molto leggera
- ✓ è sviluppata da una comunità attivissima
- ✓ è supportata da una serie di plug-in in continuo aumento, che realizzano le funzioni più richieste
- ✓ è progettata sulla base di canoni di programmazione molto moderni e per questo risulta estremamente semplice da capire e utilizzare

Iniziare ad usare JQuery



Per iniziare a usare Jquery (<http://jquery.com/>), è sufficiente importarla come script in una pagina HTML (non necessariamente dopo averla scaricata)

```
<head>
<script type="text/javascript" src="jquery.js"></script>
</head>
```

```
<head>
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
</head>
```

JQuery, come i suoi plug-in, viene distribuita in versione “normale”, utile per il debug, e “minimizzata”, molto più compatta e necessaria per rendere leggere le applicazioni finali

L'oggetto JQuery



L'interazione con JQuery avviene attraverso i metodi e le proprietà esposte dagli *oggetti JQuery*

Per esempio, per applicare un metodo JQuery a un nodo DOM o a un insieme di nodi, è necessario creare un oggetto JQuery che li rappresenti, e su questo chiamare il metodo voluto

Ogni oggetto JQuery contiene un array di elementi, possibilmente vuoto

- ✓ è possibile usare l'oggetto JQuery come un comune array, verificandone la lunghezza con la proprietà **length** e accedendo ai singoli elementi con la sintassi **[n]**

Costruire un oggetto JQuery



Gli oggetti JQuery vengono creati tramite la speciale funzione `$()`

`$ (selector) .action`

Questa funzione è molto potente e versatile, infatti può essere chiamata

Senza parametri (\$ o \$()): in questo caso, si costruisce un oggetto JQuery che non rappresenta alcun nodo DOM, ma su cui è possibile chiamare i metodi identificati da `action`

Un elemento DOM (\$ (E)): in questo caso l'oggetto conterrà (come un wrapper) l'elemento dato

Una stringa rappresentante un frammento valido di (X)HTML

`($("<p>Ciao</p>"))`: in questo caso, JQuery creerà il DOM corrispondente (senza inserirlo nel documento!)

Un stringa che rappresenta un selettore CSS (\$("a.pippo")) oppure \$("a#pluto") : in questo caso, JQuery utilizzerà gli elementi identificati dal selettore nella pagina (X)HTML per costruire l'oggetto

Primi esempi



```
var body = document.body; // l'elemento body del documento
var jqBody = $(body); // un oggetto JQuery che rappresenta l'elemento
                      // body del documento (jqBody.length == 1 && jqBody[0] == body)
var testo = "Ciao";
var html_fragment = $("<div
                      class='pippo'><p>" + testo + "</p></div>"); // un oggetto JQuery
                      // che rappresenta l'elemento div, creato (insieme al sotto-albero che parte
                      // da esso), ma non inserito nel DOM
var elemento_x = $("#x"); // un oggetto JQuery che rappresenta
                          // l'elemento con id="x" nel documento XHTM a cui lo script è associato
                          // (elemento_x[0] == document.getElementById("x"));
var paragrafi = $("p"); // un oggetto JQuery che rappresenta gli
                        // elementi con tag p nel documento (la variabile paragrafi.length è pari
                        // al numero elementi p nel documento)
```

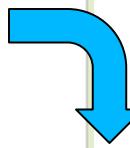


Il primo esempio JQuery



```
<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("input[type=button]").click(function(){
        $("p").hide();
    });
});
</script>
</head>

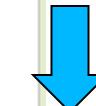
<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<form action="">
<input type="button" value="Click me"/>
</form>
</body>
</html>
```



This is a heading

This is a paragraph.

This is another paragraph.



This is a heading

La funzione **ready**



La funzione **ready** fa in modo che le action JQuery non siano eseguite prima che il documento sia completamente caricato

```
$ (document) .ready(function(){
    $("input[type=button]").click(function() {
        $("p").hide();
    });
});
```

Selettori di base JQuery (I)



I selettori utilizzabili con JQuery sono tutti quelli già visti per i Cascading StyleSheets (CSS), più altri creati ad hoc

Selettori di base

- ✓ *****
- ✓ **#id**
- ✓ **element**
- ✓ **.class**

Gerarchia

- ✓ **s1 s2** (discendenti)
- ✓ **s1 ,s2** (unione)
- ✓ **s1>s2** (figli)

Selettori di base JQuery (II)



Attributi

- ✓ **S1 [A]** (gli elementi selezionati da **S1** che hanno un attributo **A**)
- ✓ **S1 [A=v]** (gli elementi selezionati da **S1** che hanno un attributo **A** il cui valore è **v**)
- ✓ **S1 [A!=v]** (gli elementi selezionati da **S1** che hanno un attributo **A** il cui valore non è **v**)
- ✓ **S1 [A^=v]** (gli elementi selezionati da **S1** che hanno un attributo **A** il cui valore inizia con **v**)

Selettori di base JQuery (III)



Attributi

- ✓ **S1 [A\$=v]** (gli elementi selezionati da **S1** che hanno un attributo **A** il cui valore termina con **v**)
- ✓ **S1 [A*=v]** (gli elementi selezionati da **S1** che hanno un attributo **A** il cui valore ha **v** come sottostringa)
- ✓ **S1 [A~=v]** (gli elementi selezionati da **S1** che hanno un attributo **A** il cui valore contiene **v** delimitata da spazi)

Esempi



```
$("a [name]"); //oggetto JQuery che rappresenta tutti i tag a  
nel documento con attributo name
```

```
$("div.galleria > img[src$=png]"); //oggetto JQuery  
che rappresenta tutti i tag img che sono figli diretti dei div  
di classe galleria, se l'attributo src termina in png (tutte  
le immagini di tipo png contenute direttamente in div di  
classe galleria)
```

```
$("form input[type=checkbox] [checked]");  
//oggetto JQuery che rappresenta tutti i controlli di input  
inseriti in una form che hanno tipo checkbox e sono spuntati  
(hanno attributo checked); per il matching sulle form  
esistono però selettori più potenti!
```

Selettori JQuery: filtri (I)



I filtri sono pseudo classi CSS (quindi da applicare ad altri selettori, oppure ad un *隐式的) implicito)

- ✓ **S1 :eq(index)** (l'elemento n-esimo tra quelli selezionati da **S1**)
- ✓ **S1 :even** (gli elementi con indice pari tra quelli selezionati da **S1**)
- ✓ **S1 :odd** (gli elementi con indice dispari tra quelli selezionati da **S1**)

Selettori JQuery: filtri (II)



I filtri sono pseudo classi CSS (quindi da applicare ad altri selettori)

- ✓ **S1 :first** (il primo elemento tra quelli selezionati da **S1**)
- ✓ **S1 :last** (l'ultimo elemento tra quelli selezionati da **S1**)
- ✓ **S1 :gt (n)** (gli elementi con indice maggiore di **n** tra quelli selezionati da **S1**) analogo a **S1 :lt (n)**
- ✓ **S1 :not (S2)** (gli elementi selezionati da **S1** che non fanno match con **S2**)

Selettori JQuery: filtri (III)



Contenuto

- ✓ **S1 : contains (testo)** (gli elementi selezionati da **S1** che contengono il testo indicato)
- ✓ **S1 : empty** (gli elementi selezionati da **S1** che sono vuoti)
- ✓ **S1 : has (S2)** (gli elementi selezionati da **S1** che ne contengono almeno uno che soddisfa **S2**)
- ✓ **S1 : parent** (gli elementi selezionati da **S1** che hanno almeno un figlio)

Selettori JQuery: filtri (IV)



Figli

- ✓ **S1:first-child** (gli elementi selezionati da **S1** che sono il primo figlio del loro padre)
- ✓ **S1:last-child** (gli elementi selezionati da **S1** che sono l'ultimo figlio del loro padre)
- ✓ **S1:nth-child(n/even/odd)** (gli elementi selezionati da **S1** che sono il figlio **n** del loro padre, o sono i figli pari/dispari)
- ✓ **S1:only-child** (gli elementi selezionati da **S1** che sono l'unico figlio del loro padre)

Altri esempi



```
$("table tr:odd td"); // le celle contenute nelle righe dispari di tutte le tavole  
$("h2:gt(0)"); // tutte le intestazioni di livello due tranne la prima  
$("input:not([checked])"); // i controlli di input senza l'attributo checked impostato  
$("p:has(a)"); // i paragrafi che contengono almeno un link  
$("p:not(:has(a))"); // i paragrafi che non contengono un link  
$("div:contains(Saluti)"); // le div che contengono il testo "Saluti"
```

Selettori JQuery: Form (I)



JQuery definisce anche filtri specifici per gli elementi dei moduli

- ✓ **S1 :text** (gli elementi selezionati da **S1** che sono un input di tipo text)
- ✓ **S1 :checked** (gli elementi selezionati da **S1** che sono un campo spuntato)
- ✓ **S1 :disabled** (gli elementi selezionati da **S1** che sono un campo disabilitato)
- ✓ **S1 :enabled** (gli elementi selezionati da **S1** che sono un campo abilitato)
- ✓ **S1 :selected** (gli elementi selezionati da **S1** che hanno un attributo selected impostato)
- ✓ **S1 :button** (gli elementi selezionati da **S1** che sono un bottone)

Selettori JQuery: Form (II)



- ✓ **S1 : checkbox** (gli elementi selezionati da **S1** che sono una checkbox)
- ✓ **S1 : file** (gli elementi selezionati da **S1** che sono un selettore di file)
- ✓ **S1 : image** (gli elementi selezionati da **S1** che sono un input di tipo immagine)
- ✓ **S1 : input** (gli elementi selezionati da **S1** che sono un input, textarea, select o button)
- ✓ **S1 : password** (gli elementi selezionati da **S1** che sono un input di tipo password)
- ✓ **S1 : radio** (gli elementi selezionati da **S1** che sono un radio button)
- ✓ **S1 : reset** (gli elementi selezionati da **S1** che sono un bottone di reset)
- ✓ **S1 : submit** (gli elementi selezionati da **S1** che sono un bottone di submit)

Esempi



```
$(".small:text"); // gli input di tipo testuale di classe  
small  
$("#selector option:selected"); // l'opzione  
selezionata nell'elemento (select) con id="selector"  
$("input[type=checkbox]:not(:checked)"); // gli  
elementi input di tipo checkbox non spuntati  
$("#form1 :submit"); // i bottoni di submit nell'elemento  
(form) con id="form1"
```

Modifica di contenuto e attributi (I)



- ✓ JQuery mette a disposizione molti metodi standard per manipolare il contenuto degli elementi in maniera sicura e *crossbrowser*
- ✓ Va notato che, se applicati su oggetti JQuery che rappresentano più di un elemento, i metodi di lettura restituiscono il valore estratto dal primo elemento dell'insieme, mentre quelli di impostazione agiscono su ciascun elemento dell'insieme stesso

Modifica di contenuto e attributi (II)



- ✓ `attr(A)` restituisce il valore dell'attributo **A**
- ✓ `attr(A,V)` imposta l'attributo **A** al valore **V**
- ✓ `removeAttr(A)` rimuove l'attributo **A**
- ✓ `html()` restituisce il codice html contenuto nell'elemento
- ✓ `html(T)` imposta il codice html **T** come contenuto dell'elemento
- ✓ `text()` restituisce il testo contenuto nell'elemento (eliminando l'eventuale markup)
- ✓ `text(T)` imposta il testo **T** come contenuto dell'elemento
- ✓ `val()` restituisce il valore di un controllo di form
- ✓ `val(V)` imposta a **V** il valore di un controllo di form

Esempi



```
$("#risultato").html("<p>Nessun riscontro</p>"); // inserisce il  
paragrafo dato all'interno dell'elemento con id=risultato  
$("#risultato").text(); // restituisce il solo testo contenuto  
nell'elemento; in questo esempio "Nessun riscontro"  
$(":input").val(); // restituisce il value del primo controllo input,  
textarea, select o button nel documento  
$("#pippo").val("pluto"); // se l'elemento con id=pippo è un input  
testuale o una textarea, ne sostituisce il contenuto con la stringa "pluto";  
se l'elemento è una select, ne seleziona l'opzione avente come valore  
"pluto" (se esiste); negli altri casi, non ha alcun effetto  
$(document.body).attr("lang","it"); // importa a "it" l'attributo  
"lang" sull'elemento body del documento  
$(".lang").removeAttr("lang"); // rimuove l'attributo "lang" da tutti  
gli elementi che lo specificano
```

Manipolazione del DOM (I)



JQuery fornisce metodi di manipolazione del DOM

- ✓ **append (C)** accoda **C** ai figli degli elementi nell'insieme a cui il metodo è applicato; **C** può essere un elemento DOM, una stringa HTML e un oggetto JQuery (che rappresenta uno o più elementi)
- ✓ **appendTo (S)** accoda gli elementi dell'insieme a cui il metodo è applicato all'insieme dei figli degli elementi selezionati tramite **S**, che può essere un elemento DOM, un selettore CSS, una stringa HTML o un oggetto JQuery
- ✓ **prepend (C)/prependTo (S)** funzionano come **append** e **appendTo**, ma inseriscono all'inizio della lista dei figli

Manipolazione del DOM (II)



- ✓ **after (C) /before (C)** inseriscono **C** prima/dopo gli elementi dell'insieme a cui il metodo è applicato
- ✓ **insertAfter (S) /insertBefore (S)** inseriscono gli elementi dell'insieme a cui il metodo è applicato prima/dopo quelli selezionati da **S**
- ✓ **empty ()** elimina il contenuto di tutti gli elementi a cui è applicato
- ✓ **remove ()** rimuove tutti gli elementi a cui è applicato
- ✓ **detach ()** rimuove tutti gli elementi a cui è applicato, ma non li distrugge, in modo che possano essere reinseriti altrove
- ✓ **clone (B)** esegue una copia completa degli elementi a cui è applicato; se **B** è true, vengono copiati anche gli **event handlers** associati all'elemento da JQuery

Esempi



```
$("div.translation") .append("<p>Powered by me</p>"); //  
    inserisce il paragrafo dato (dopo averlo creato) alla fine delle div con  
    classe translation  
  
$("#a > li") .appendTo("#b"); // sposta (nel DOM inserire un elemento  
    già presente in un'altra locazione corrisponde a spostarlo) tutti gli item della  
    lista con id="a" alla fine della lista con id="b"  
  
var frammento = $("<p>pippo</p>"); // crea un frammento html  
frammento.appendTo("#a"); // appende il frammento all'elemento con  
    id="a"  
  
frammento.clone() .prependTo("div.marked:first"); //inserisce  
    una copia del frammento all'inizio della prima div di classe marked  
  
var exel = $("p:first") .detach(); // rimuove ma non cancella il  
    primo paragrafo del documento...  
  
exel.appendTo(document.body); // e lo reinserisce alla fine del  
    documento
```

Le espressioni regolari (I)



Le espressioni regolari sono molto utili per eseguire il **parsing e la validazione dei dati** immessi dall'utente

Javascript (e quindi anche jQuery) riconoscono le espressioni regolari scritte nella **sintassi Perl**

Un'espressione regolare **costante** è definita dalla seguente sintassi

/pattern/modifiers

Un'espressione regolare può anche essere generata a tempo di esecuzione tramite il costruttore **RegExp**

```
regularExpr = new RegExp(pattern,modifiers)
```

Le espressioni regolari (II)



Modifiers	Descrizione
i	Effettua un matching case-insensitive
g	Trova tutti i match anziché fermarsi alla prima occorrenza
Espressione	Descrizione
[abc]	Caratteri uguali a uno di quelli tra parentesi quadre
[^abc]	Caratteri diversi da quelli tra parentesi quadre
[0-9]	Qualsiasi cifra tra 0 e 9
[A-Z]	Qualsiasi lettera tra A e Z (maiuscole)
[a-z]	Qualsiasi lettera tra a e z (minuscole)
[A-z]	Qualsiasi lettera tra A (maiuscola) e z (minuscola)
(red green blue)	Una delle espressioni alternative elencate

Le espressioni regolari (III)



Metacarattere	Descrizione
.	Carattere singolo, eccetto newline o carattere di fine riga
\w	Una lettera qualsiasi
\W	Un carattere che non sia una lettera
\d	Una cifra qualsiasi
\D	Un carattere che non sia una cifra
\s	Un carattere non visibile, come uno spazio, una tabulazione o un fine riga
\S	Un carattere che non sia tra quelli non visibili
\n \r \t	New line, carriage return, tab
\xxx	Carattere specificato dal numero in notazione ottale xxx
\xdd	Carattere specificato dal codice esadecimale dd

Le espressioni regolari (IV)



Quantificatori	Descrizione
n+	Una stringa di uno o più “n”
n*	Una stringa con zero o più “n”
n?	Una stringa con uno o nessun “n”
n{X}	Una stringa che contiene X caratteri “n”
n{X,Y}	Una stringa che contiene da X a Y caratteri “n”
n{X,}	Una stringa che contiene almeno X caratteri “n”
n\$	Una stringa che termina con il carattere “n”
^n	Una stringa che inizia con il carattere “n”
?=n	Una stringa seguita dal carattere “n”
?!n	Una stringa non seguita dal carattere “n”

Le espressioni regolari (V)



È possibile utilizzare le espressioni regolari con vari metodi da invocare sull'oggetto stesso che rappresenta l'espressione **r**

- ✓ **r.test(s)** restituisce **true** se la stringa **s** è conforme all'espressione regolare **r**
- ✓ **s.match(r)** restituisce un'array con tutte le corrispondenze dell'espressione regolare **r** trovate nella stringa **s**
- ✓ **s.replace(r, sNew)** individua nella stringa **s** tutte le corrispondenze con l'espressione **r** e le sostituisce con **sNew**
- ✓ **s.split(r)** divide la stringa **s** in una serie di segmenti distinti dai separatori specificati con l'espressione **r** e li restituisce come array
- ✓ **s.search(r)** cerca nella stringa **s** la prima corrispondenza con l'espressione regolare **r** e restituisce l'indice della posizione trovata



PROGRAMMAZIONE AD EVENTI

P
W

La programmazione ad eventi



- ✓ Javascript è un linguaggio fortemente *orientato agli eventi*
 - ✓ l'utente interagisce con la pagina muovendo il mouse, cliccando, digitando qualcosa in una casella di testo, etc.
 - ✓ la pagina deve “reagire” opportunamente a queste sollecitazioni
 - ✓ il gestore dell'evento (*event handler*) è una funzione che risponde alla “mossa” dell'utente
 - ✓ è possibile catturare gli eventi e gestirli
- ✓ Gli eventi possono essere generati dall'utente attraverso le proprie azioni sul documento o possono essere generati dal browser (e.g., caricamento delle pagine, caricamento del documento)

Eventi in Javascript



- Abort
- Blur
- Change
- Click
- DblClick
- Error
- Focus
- KeyDown
- KeyPress
- KeyUp
- Load
- MouseDown
- MouseMove
- MouseOut
- MouseOver
- MouseUp
- Move
- Reset
- Resize
- Select
- Submit
- Unload

Gestore degli eventi



- ✓ Ad ogni evento viene associato un gestore dell'evento
- ✓ Per ottenere il nome del gestore di un evento basta aggiungere il prefisso **on** al nome dell'evento stesso
- ✓ Il gestore di un evento permette di associare ad un evento
 - ✓ La semplice invocazione di una funzione
 - ✓ Una sequenza di comandi complessi separati da un punto e virgola

```
<input type="button" value="Hello" onclick="alert('Hello World!');" />
```

Esempi di gestori degli eventi



Click → `onclick`

- attivato quando l'utente clicca su un elemento

MouseOver → `onmouseover`

- attivato quando il cursore del mouse si muove sopra un elemento

MouseOut → `onmouseout`

- attivato quando il cursore del mouse si sposta fuori un elemento

Gestione degli eventi



JQuery dispone di una gestione degli eventi che permette di superare molte delle incompatibilità tra i browser

Le funzionalità di *event handling* sono accessibili tramite il metodo **on**:

on (T, F) aggancia, negli elementi dell'insieme, all'evento specificato da **T** l'*event handler* **F**

- ✓ **T** è una stringa contenente il nome di un evento Javascript (ad es. "click")
- ✓ **F** è una funzione che verrà chiamata assegnando al suo contesto (**this**) l'elemento che ha scatenato l'evento e passando l'oggetto evento come unico argomento

off (T, F) rimuove **F** dalla lista degli *handlers* per l'evento **T** negli elementi dell'insieme

Gestione degli eventi: shortcut



Anche in JQuery esistono dei metodi-scorciatoia per eseguire il binding diretto di eventi, ad esempio **click (F)**

`$ (selector) .click(function)`

Aziona la funzione al click del mouse sull'elemento selezionato

`$ (selector) .dblclick(function)`

Aziona la funzione al doppio click del mouse sull'elemento selezionato

`$ (selector) .mouseover(function)`

Aziona la funzione quando il mouse si sposta sull'elemento selezionato

Esempio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html>  
    <head>  
        <title>JQuery – Binding di Event Handlers</title>  
        <script type="text/javascript" src="https://code.jquery.com/jquery.js"></script>  
  
        <style>  
            p {  
                background: yellow;  
                font-weight: bold;  
                cursor: pointer;  
                padding: 5px;  
            }  
            span {  
                color: red;  
            }  
        </style>  
    </head>  
    <body>  
        <p>  
            Click or double click here.  
        </p>  
        <span></span>  
        <script>  
            $("p").on("click", function(event) {  
                var str = "(" + event.pageX + ", " + event.pageY + ")";  
                $("span").text("Click happened! " + str);  
            });  
            $("p").on("dblclick", function() {  
                $("span").text("Double-click happened in " + this.nodeName);  
            });  
        </script>  
    </body>  
</html>
```



ANIMAZIONI



Animazioni di base



JQuery dispone di diverse funzioni per animare gli elementi

- ✓ **hide (T ,C) /show (T ,C)** nasconde/mostra ciascun elemento dell'insieme; se **T** è fornito, gli elementi vengono animati fino a scomparire/apparire in **T** millisecondi; se **C** è fornito, è una funzione che viene chiamata appena gli elementi sono scomparsi/apparsi (*funzione di callback*)
- ✓ **fadeOut (T ,C) /fadeIn (T ,C)** agiscono come **hide** e **show**, ma con un effetto sfumato
- ✓ **slideUp (T ,C) /slideDown (T ,C)** agiscono come **hide** e **show**, ma con un effetto di scorrimento
- ✓ **animate (style ,T ,C)** modifica le proprietà CSS specificate in **style**, nel tempo **T** (se specificato, può assumere anche i valori **fast**, **slow** e **normal**); **C** è la funzione di callback
- ✓ Il filtro **S1 :animated** consente di filtrare gli elementi selezionati da S1 che sono animati

Esempio



// aggiorna in maniera “dolce” il contenuto dell’elemento con **id=display**:
prima fa scomparire in maniera sfumata l’elemento, quindi appena
quest’ultimo è invisibile, ne modifica il contenuto html e lo rende di nuovo
visibile, sempre in modo sfumato

```
var dis = $("#display");
dis.fadeOut(1000, function() {
    dis.html("<b>Sorpresa!</b>").fadeIn(1000);
});
```

Esempio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <title>JQuery - Animazioni di base</title>
    <script type="text/javascript" src="https://code.jquery.com/jquery.js"></script>
    <script type="text/javascript">
      $(document).ready(function() {
        $("button").click(function() {
          $("div").animate({
            height : 300
          }, "slow");
          $("div").animate({
            width : 300
          }, "slow");
          $("div").animate({
            height : 100
          }, "slow");
          $("div").animate({
            width : 100
          }, "slow");
        });
      });
    </script>
  </head>

  <body>
    <button>
      Start Animation
    </button>
    <br />
    <br />
    <div style="background: #98bf21; height:100px; width:100px; position: relative"> </div>
  </body>
</html>
```



Manipolazione dello stile



JQuery dispone di tre metodi per manipolare l'attributo di classe degli elementi:

- ✓ **addClass (C)** aggiunge la classe **C** (attributo **class** html)
- ✓ **removeClass (C)** rimuove la classe **C**
- ✓ **hasClass (C)** vero se almeno un elemento dell'insieme ha classe **C**

È inoltre possibile manipolare direttamente gli stili di ogni elemento utilizzando le funzioni

- ✓ **css (P)** restituisce il valore corrente (calcolato) della proprietà CSS, specificata secondo lo standard W3C
- ✓ **css (P ,V)** imposta la proprietà CSS **P** al valore **V**

Esempio



```
<html>
<head>
  <style>
    p { margin: 8px; font-size:16px; }
    .selected { color:red; }
    .highlight { background:yellow; }
  </style>
  <script src="http://code.jquery.com/jquery-1.5.js"></script>
</head>
<body>
  <p>Hello</p>
  <p>and</p>
  <p>Goodbye</p>
<script>
  $("p:last").addClass("selected highlight");
</script>
</body>
</html>
```

Hello
and
Goodbye

Esempio

```
<html>
  <head>
    <title>JQuery - Animazioni di base</title>
    <script type="text/javascript" src="https://code.jquery.com/jquery.js"></script>
    <style>
      div {
        background: yellow;
        border: 1px solid #AAA;
        width: 80px;
        height: 80px;
        margin: 0 5px;
        float: left;
      }
      div.colored {
        background: green;
      }
    </style>
  </head>
  <body>
    <form>
      <input type="button" id="run" value="Run">
    </form>
    <div> </div>
    <div id="mover"> </div>
    <div> </div>
    <script>
      $("#run").click(function() {
        $("div:animated").toggleClass("colored");
      });
      function animateIt() {
        $("#mover").slideToggle("slow", animateIt);
      }

      animateIt();
    </script>
  </body>
</html>
```

Iterazione su un oggetto JQuery



Per eseguire operazioni complesse, è possibile iterare l'applicazione di una funzione su tutti gli elementi dell'insieme contenuto in un oggetto JQuery usando il metodo **each**

each (F), dove **F** è una funzione, richiama **F** una volta per ogni elemento dell'insieme, impostandone il contesto (**this**) all'elemento corrente e passando opzionalmente come argomento l'indice dell'elemento nell'insieme

Esempi



// il codice che segue imposta su ciascuna **div** del documento un handler per il click che visualizza un messaggio contenente il numero d'ordine della **div** nel documento

```
$("div").each(function(i) {  
    $(this).on("click",function(e) {  
        alert("Questa è la DIV numero "+i);  
    });  
});
```

// il frammento che segue popola l'array **colors** con i colori di tutti i paragrafi nel documento

```
var colors = [];  
$("p").each(function() {  
    colors.push($(this).css("color"));  
});
```



Altri esempi



Sulla pagina Moodle del corso:

- ✓ Orologio digitale
- ✓ Conto alla rovescia
- ✓ Timer
- ✓ Validazione input utente
- ✓ Interazione con i controlli di una form (e.g., focus, submit)
- ✓ Interazioni tramite il mouse

Tutorial W3School

- ✓ <https://www.w3schools.com/jquery/default.asp>



PROGRAMMAZIONE WEB

SCRIPTING CLIENT-SIDE IN JAVASCRIPT LA LIBRERIA JQUERY

Prof. Ada Bagozi

ada.bagozi@unibs.it





PROGRAMMAZIONE WEB

LA TECNOLOGIA AJAX

Prof. Ada Bagozi

ada.bagozi@unibs.it



Un nuovo modello



- ✓ L'utilizzo di DHTML (JavaScript/Eventi + DOM + CSS) delinea un nuovo modello per le applicazioni Web
- ✓ In pratica ci troviamo di fronte ad un modello ad eventi simile a quello delle applicazioni tradizionali
- ✓ Abbiamo però due livelli di eventi:
 - ✓ **eventi locali** che sono gestiti da event handler codificati in Javascript e che portano ad un cambiamento locale della pagina
 - ✓ **eventi remoti** che sono gestiti tramite ricaricamento della pagina che viene modificata sul lato server in base ai parametri passati tramite HTTP
- ✓ Il ricaricamento di una pagina per rispondere ad un evento remoto prende il nome di **postback**

Esempio di evento remoto (I)



- ✓ Consideriamo un modulo in cui compaiono due menu a tendina che servono a selezionare il comune di nascita di una persona
 - ✓ un menu con le province
 - ✓ un menu con i comuni
- ✓ Si vuole fare in modo che, scegliendo la provincia nel primo menu a tendina, appaiano nel secondo menu solo i comuni di quella provincia

Esempio di evento remoto (II)



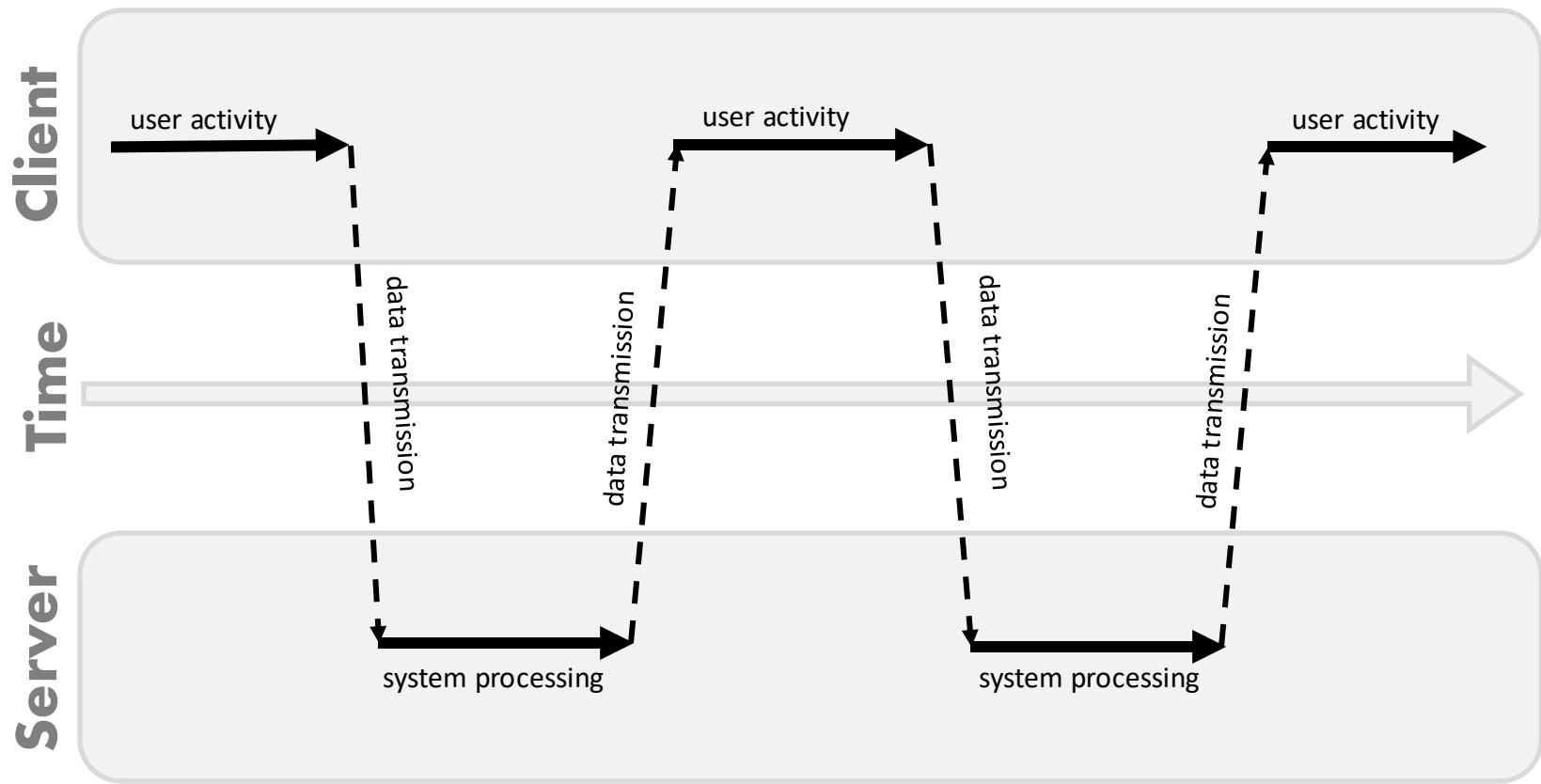
- ✓ Per realizzare questa interazione si crea uno script lato server (per esempio, in PHP) che inserisce nel menu a tendina dei comuni l'elenco di quelli che appartengono alla provincia passata come parametro
 1. si definisce un evento **onChange** collegato all'elemento **select** delle province che forza il ricaricamento della pagina (**postback**) quando una provincia viene selezionata
 2. l'utente sceglie una provincia
 3. viene invocato lo script lato server con il parametro della provincia impostato al valore scelto dall'utente
 4. la pagina restituita contiene nel menu a tendina dei comuni l'elenco di quelli che appartengono alla provincia scelta

Limiti del modello



- ✓ Quando lavoriamo con applicazioni desktop siamo abituati ad un elevato livello di interattività:
 - ✓ le applicazioni reagiscono in modo rapido ed intuitivo ai comandi
- ✓ Le applicazioni Web tradizionali espongono invece un modello di interazione rigido
 - ✓ modello “Click, wait and refresh”
 - ✓ è necessario il refresh della pagina da parte del server per la gestione di qualunque evento remoto (sottomissione di dati tramite form, visita di un link per ottenere informazioni di interesse, ...)
- ✓ È ancora un **modello sincrono**: l’utente effettua una richiesta e deve attendere la risposta da parte del server

Modello di interazione classico



AJAX (I)



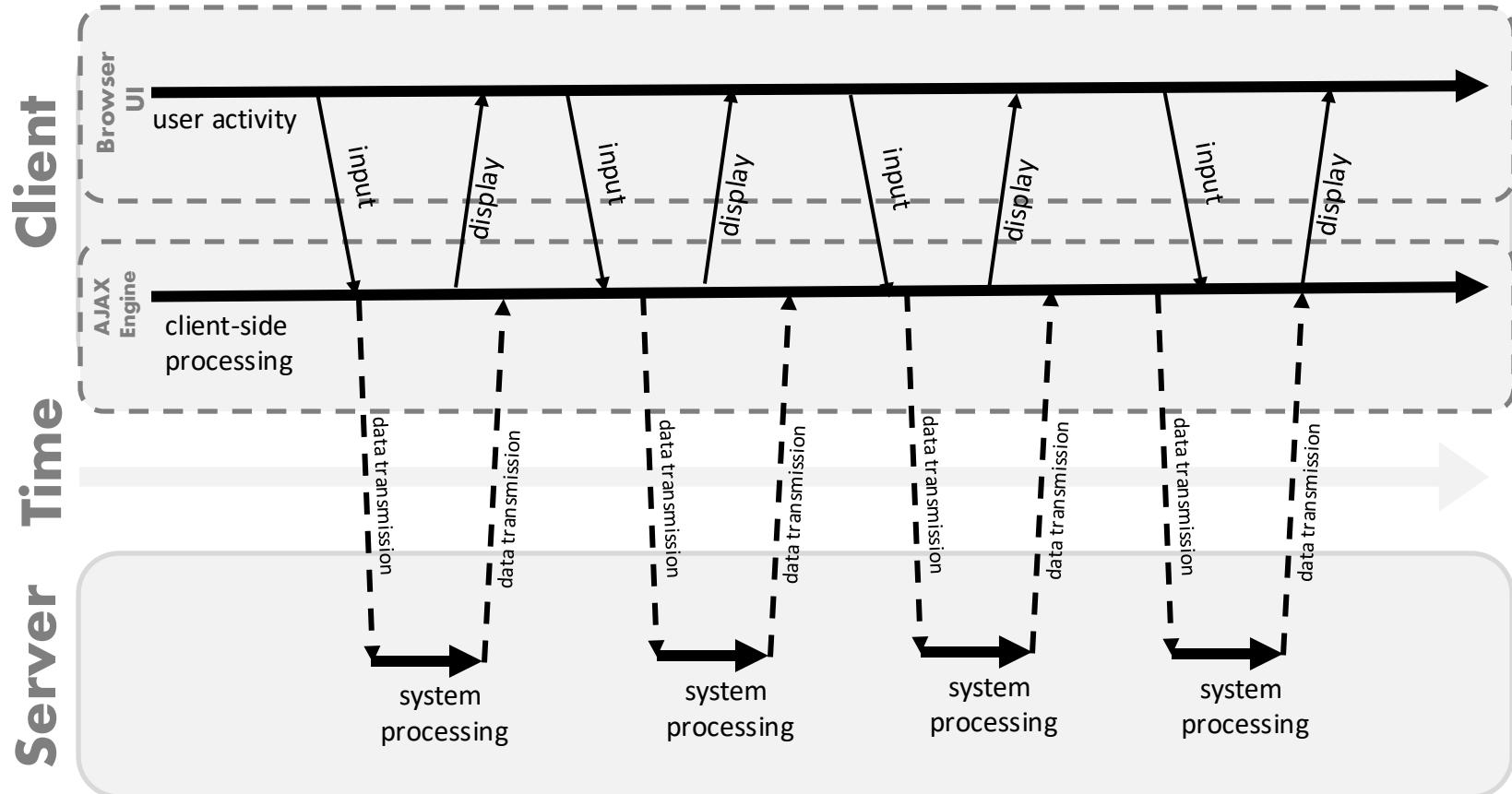
- ✓ Il **modello AJAX** è nato per superare queste limitazioni
- ✓ AJAX non è un acronimo ma spesso viene interpretato come **Asynchronous Javascript And XML**
- ✓ È basato su tecnologie standard:
 - ✓ JavaScript (e jQuery)
 - ✓ DOM
 - ✓ XML/JSON
 - ✓ HTML
 - ✓ CSS

AJAX (II)



- ✓ AJAX punta a supportare applicazioni user friendly con un'interattività elevata
- ✓ Per tali applicazioni, si usa spesso il termine **RIA (Rich Internet Application)**
- ✓ L'idea alla base di AJAX è quella di consentire agli script JavaScript di interagire direttamente con il server
 - ✓ ottenere dati dal server senza la necessità di ricaricare l'intera pagina
 - ✓ introdurre anche una *comunicazione asincrona* fra client e server: il client non interrompe l'interazione con l'utente anche quando è in attesa di risposte dal server

Modello di interazione classico



Tipica sequenza AJAX



- ✓ Si verifica un evento determinato dall'interazione fra l'utente e la pagina Web
- ✓ L'evento comporta l'esecuzione di una funzione JavaScript in cui:
 - ✓ si prepara l'invio ad un URL sul server, predisponendo delle funzioni di **callback**, da eseguire all'arrivo della risposta del server
 - ✓ si effettua una chiamata al server
- ✓ Il server elabora la richiesta e risponde al client
- ✓ Il browser invoca l'azione asincrona che:
 - ✓ elabora il risultato
 - ✓ aggiorna il DOM della pagina per mostrare i risultati dell'elaborazione

La chiamata AJAX



- ✓ La chiamata AJAX effettua la richiesta di una risorsa via HTTP ad un server Web
 - ✓ non sostituisce l'URI della propria richiesta all'URI corrente
 - ✓ non provoca un cambio di pagina
 - ✓ può inviare eventuali informazioni (parametri) sotto forma di variabili (come un modulo)
 - ✓ può effettuare sia richieste GET che POST
- ✓ Le richieste possono essere di tipo
 - ✓ **sincrono**: blocca il flusso di esecuzione del codice Javascript (ci interessa poco)
 - ✓ **asincrono**: non interrompe il flusso di esecuzione del codice Javascript né le operazioni dell'utente sulla pagina

Gli ingredienti di AJAX



- ✓ L'URL della risorsa remota di invocare
 - ✓ tipicamente uno script, che ritorna (parte di) una pagina HTML, ma anche del semplice testo (plain, JSON, XML)
- ✓ Gli eventuali dati da passare allo script eseguito in remoto
- ✓ Il metodo HTTP da utilizzare (e, se necessario, gli header HTTP da impostare)
- ✓ La modalità (**asincrona**, sincrona)
- ✓ Il tempo che si è disposti ad aspettare per l'arrivo della risposta
- ✓ Le funzioni (**callback**) da invocare, in caso di successo o insuccesso della richiesta, di errore etc.

JQuery e AJAX (I)



JQuery rende l'uso di AJAX estremamente rapido e agevole

`$.ajax(url[, settings])` esegue una chiamata AJAX all'indirizzo `url`, configurandola opportunamente attraverso l'insieme di coppie `chiave, valore` contenute nell'oggetto `settings`

- `async` – un parametro booleano per stabilire se la chiamata AJAX è sincrona (bloccante) o asincrona (opzione di default – `{async: true}`)
- `data` – i dati da inviare lato server per processare la richiesta (*query string*); a sua volta, questo parametro è impostato come un insieme di coppie `chiave, valore`
- `headers` – un insieme di coppie `chiave, valore` per settare l'header della richiesta HTTP che si sta inviando al server
- `method` – il metodo HTTP da utilizzare per inviare la richiesta al server («get», «post») – il valore di default è «get»

JQuery e AJAX (II)



- **dataType** – il formato atteso della risposta HTTP da parte del server («xml», «html», «json», «text»)
- **statusCode** – un oggetto contenente i codici HTTP della risposta (p.e., «200» OK) e una funzione per ciascun codice da eseguirsi al verificarsi del codice stesso

```
1 | $.ajax({  
2 |   statusCode: {  
3 |     404: function() {  
4 |       alert( "page not found" );  
5 |     }  
6 |   }  
7 |});
```

- **timeout** – un valore espresso in millisecondi per fermare la gestione della richiesta



JQuery e AJAX – Callback functions



- **success** – una funzione che viene invocata quando la richiesta è terminata con successo; gli argomenti della funzione sono il contenuto della risposta, formattata secondo il **dataType**, una stringa che rappresenta lo stato della risposta e un oggetto di tipo **xmlHttpRequest**
- **error** – una funzione che viene invocata quando la richiesta fallisce; gli argomenti della funzione sono un oggetto di tipo **xmlHttpRequest**, una stringa che rappresenta lo stato della risposta e una stringa contenente il messaggio di errore
- **complete** – una funzione che viene invocata quando la richiesta è stata gestita (dopo l'esecuzione di **success** e **error**); gli argomenti della funzione sono un oggetto di tipo **xmlHttpRequest** e una stringa che rappresenta lo stato della risposta («success», «nocontent», «error», «timeout»)



Deprecation notice – Da jQuery3.0 **complete**, **error** e **success** sono sostituite rispettivamente da **always**, **fail** e **done**

Proprietà di XMLHttpRequest



- ✓ Stato e risultati della richiesta vengono memorizzati dall'interprete Javascript all'interno dell'oggetto **XMLHttpRequest** durante la sua esecuzione
- ✓ Per compatibilità all'indietro, **XMLHttpRequest** presenta le proprietà seguenti:
readyState
status
statusText
statusCode
responseText
responseXML

Proprietà ReadyState



- ✓ Proprietà in sola lettura di tipo intero che consente di leggere in ogni momento lo stato della richiesta; ammette 5 valori:
 - 0: **uninitialized** - l'oggetto esiste, ma non è stato invocato il metodo **open()**
 - 1: **open** - è stato invocato il metodo **open()**, ma **send()** non ha ancora effettuato l'invio dati
 - 2: **sent** - il metodo **send()** è stato eseguito ed ha inviato la richiesta
 - 3: **receiving** – la risposta ha iniziato ad arrivare
 - 4: **loaded** - l'operazione è stata completata
- ✓ Attenzione:
 - ✓ questo ordine non è sempre identico e non è sfruttabile allo stesso modo su tutti i browser
 - ✓ l'unico stato supportato da tutti i browser è il 4

Metodi di XMLHttpRequest



I metodi `getAllResponseHeaders()` e `getResponseHeader(name)` consentono di leggere gli header HTTP che descrivono la risposta del server

- ✓ sono utilizzabili solo in caso di ricezione della risposta (`readyState>=3`)

Il metodo `setRequestHeader(nomeheader, valore)` consente di impostare gli header HTTP della richiesta da inviare

- ✓ viene invocata più volte, una per ogni header da impostare
- ✓ per una richiesta GET gli header sono opzionali
- ✓ sono invece necessari per impostare la codifica utilizzata nelle richieste POST

Esempio (I)



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
|   "http://www.w3.org/TR/html4/strict.dtd">
<html>
|   <head>
|       <title>.: Un semplice esempio AJAX :.</title>
|       <script type="text/javascript" src="https://code.jquery.com/jquery.js"></script>
|   </head>
|   <body>
|       <h1>Ricerca nome</h1>
|       <form>
|           <p>Cerca un nome per il tuo bambino: noi abbiamo centinaia di suggerimenti da darti!<p>
|           <input type="text">
|       </form>
|
|       <div id="risultati"></div>
|       <script>
|           $(document).ready(function(){
|               $("input[type=text]").keyup(function(){
|                   $.ajax("ajax.php", {data: {nome: $(this).val()}}, success: function(result) {
|                       $("#risultati").html(result);
|                   });
|               });
|           });
|       </script>
|   </body>
|</html>
```

```
<?php  
    $nomi[0] = "Alessandro";  
    $nomi[1] = "Alessio";  
    $nomi[2] = "Claudio";  
    $nomi[3] = "Davide";  
    $nomi[4] = "Dario";  
    $nomi[5] = "Francesco";  
    $nomi[6] = "Giancarlo";  
    $nomi[7] = "Luca";  
    $nomi[8] = "Luigi";  
  
    $nome = $_GET["nome"];  
    $risultato = "";  
  
    if (strlen($nome) > 0) {  
        for ($i = 0; $i < count($nomi); $i++) {  
            if (strtoupper($nome) == strtoupper(substr($nomi[$i], 0, strlen($nome)))) {  
                if ($risultato == "") {  
                    $risultato = $nomi[$i];  
                }  
                else {  
                    $risultato .= ", " . $nomi[$i];  
                }  
            }  
        }  
        if ($risultato == "") {  
            echo "Nessun risultato...";  
        }  
        else {  
            echo $risultato;  
        }  
    }  
?>
```

ajax.php?nome=...



AJAX e XML



- ✓ Spesso i dati scambiati fra client e server sono codificati in XML
- ✓ AJAX come abbiamo visto è in grado di ricevere documenti XML
- ✓ In particolare è possibile elaborare i documenti XML ricevuti utilizzando le API W3C DOM
- ✓ Per visualizzare i contenuti ricevuti modifichiamo il DOM della pagina HTML

AJAX e JSON (I)



- ✓ JSON è tuttavia il formato più utilizzato per lavorare con AJAX
- ✓ Sul lato client:
 - ✓ si crea un oggetto JavaScript e si riempiono le sue proprietà con le informazioni necessarie
 - ✓ si usa **JSON.stringify()** per convertire l'oggetto in una stringa JSON
 - ✓ si manda la stringa al server mediante **XMLHttpRequest** (la stringa viene passata come variabile con GET o POST)

AJAX e JSON (II)



- ✓ Sul lato server:
 - ✓ si decodifica la stringa JSON e la si trasforma utilizzando un apposito parser (si trova sempre su www.json.org)
 - ✓ si elabora l'oggetto
 - ✓ si crea un nuovo oggetto JSON che contiene i dati della risposta
 - ✓ si trasmette la stringa JSON al client nel corpo della risposta HTTP

AJAX e JSON (III)



- ✓ Sul lato client:
 - ✓ si converte la stringa JSON in un oggetto Javascript usando **JSON.parse()**
 - ✓ si usa liberamente l'oggetto per gli scopi desiderati

AJAX e JQuery - Shortcuts



`$.get(url [, data] [, success], [, dataType])` esegue una chiamata al server asincrona di tipo **GET**

`$.post(url [, data] [, success], [, dataType])` esegue una chiamata al server asincrona di tipo **POST**

`$.getJSON(url [, data] [, success])` esegue una chiamata al server asincrona di tipo **GET** e restituisce un risultato in formato **JSON**

<https://api.jquery.com/jquery.ajax/>

<https://api.jquery.com/category/ajax/shorthand-methods/>

```
<head>
    <title>Esempio nell'uso di JSON</title>
    <script type="text/javascript" src="lib/json2.js"></script>
    <script type="text/javascript" src="https://code.jquery.com/jquery.js"></script>
</head>

<body>
    <form name="personal" action="" method="POST">
        Nome <input type="text" name="firstname"><br>
        Email <input type="text" name="email"><br>
        Hobby <input type="checkbox" name="hobby" value="sport"> Sport
            <input type="checkbox" name="hobby" value="lettura"> Lettura
            <input type="checkbox" name="hobby" value="musica"> Musica
        <input type="button" name="valid" value="Validate">
        <script>
            $(document).ready(function(){
                $("input[type=button]").click(function() {
                    var JSONObject = new Object;
                    JSONObject.firstname = $("input[name=firstname]").val();
                    JSONObject.email = $("input[name=email]").val();
                    JSONObject.hobby = new Array;

                    $("input[type=checkbox]:checked").each(function(i){
                        JSONObject.hobby[i] = new Object;
                        JSONObject.hobby[i].hobbyName = $(this).val();
                    });

                    JSONstring = JSON.stringify(JSONObject);
                    alert(JSONstring);
                    $.ajax("parser.php", {data: {json: JSONstring}, success: function(result){
                        alert(result);
                    }});
                });
            });
        </script>
    </form>
</body>
```



Esempio



```
<?php  
// decodifica della stringa JSON in un oggetto PHP  
$decoded = json_decode($_GET['json']);  
  
// creazione della risposta sotto forma di oggetto  
$json = array();  
$json['name'] = $decoded->firstname;  
$json['email'] = $decoded->email;  
$json['hobbies'] = array();  
for($i=0; $i<count($decoded->hobby); $i++)  
{  
| $json['hobbies'][] = $decoded->hobby[$i]->hobbyName;  
}  
  
// codifica dell'array $json in una stringa JSON  
$encoded = json_encode($json);  
  
// invio della risposta e fine dello script  
die($encoded);  
?>
```

parser.php?json=...

Vantaggi e svantaggi di AJAX



- ✓ Si guadagna in espressività, ma si perde la linearità dell'interazione
- ✓ Mentre l'utente è all'interno della stessa pagina le richieste sul server possono essere numerose e indipendenti

- ❖ Il tempo di attesa passa in secondo piano o non è avvertito affatto
- ❖ Possibili criticità sia per l'utente che per lo sviluppatore
 - ❖ percezione che non stia accadendo nulla (sito che non risponde)
 - ❖ problemi nel gestire un modello di elaborazione che ha bisogno di aspettare i risultati delle richieste precedenti

Criticità nell'interazione con l'utente



- ✓ Le richieste AJAX permettono all'utente di continuare a interagire con la pagina
- ✓ Ma non necessariamente lo informano di cosa sta succedendo e possono durare troppo!
- ✓ L'effetto è un disorientamento dell'utente
- ✓ Dobbiamo quindi agire su due fronti:
 - ✓ rendere visibile in qualche modo l'andamento della chiamata (barre di scorrimento ecc.)
 - ✓ interrompere le richieste che non terminano in tempo utile per sovraccarichi del server o momentanei problemi di rete (timeout)

Aspetti critici per il programmatore



- ✓ È accresciuta la complessità delle applicazioni Web
- ✓ La logica di presentazione è ripartita fra client-side e server-side
- ✓ Le applicazioni AJAX pongono problemi di debug, test e mantenimento
- ✓ Il test di codice JavaScript è complesso
- ✓ Il codice JavaScript ha problemi di modularità
- ✓ I toolkit AJAX sono molteplici e solo recentemente hanno raggiunto una discreta maturità
- ✓ Mancanza di standardizzazione nei vecchi browser

FETCH



È l'API nativa di JavaScript per effettuare richieste HTTP.

- ❖ Sostituisce metodi più vecchi come XMLHttpRequest e \$.ajax() di jQuery
- ❖ Basata su Promesse, quindi si integra perfettamente con `async/await`
- ❖ Supportata da tutti i browser moderni (tranne Internet Explorer)

FETCH vs AJAX



Feature	\$.ajax() (jQuery)	fetch() (Native)
Built-in	✗ Needs jQuery	✓ Yes, in modern browsers
Promise-based	✗ (callback-based)	✓ Native support
Automatic JSON parse	✓ (with dataType)	✗ You must call .json()
HTTP error thrown	✓ Triggers error	✗ Must manually check response.ok
Cancel requests	✓ With .abort()	✓ With AbortController
CORS cookies	✓ Easier	✗ Must use credentials flag

FETCH Examples



Esempio Base

```
fetch('/api/dati')
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.error('Errore:', error));
```

Esempio con async/await

```
async function caricaDati() {
    try {
        const risposta = await fetch('/api/dati');
        const dati = await risposta.json();
        console.log(dati);
    } catch (errore) {
        console.error('Errore nella richiesta:', errore);
    }
}
```





PROGRAMMAZIONE WEB

LA TECNOLOGIA AJAX

Prof. Ada Bagozi

ada.bagozi@unibs.it





PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - PYTHON

Prof. Ada Bagozi

ada.bagozi@unibs.it



Cos'è il Python?



- ✓ *Python* è un linguaggio di programmazione ad alto livello, interpretato e dinamico
- ✓ Creato da **Guido van Rossum** e rilasciato nel 1991
- ✓ Alcune caratteristiche “tecniche”
 - ✓ è un **linguaggio di scripting**
 - ✓ è **interpretato** (non compilato)
 - ✓ è **multipiattaforma**
 - ✓ è **tipizzato dinamicamente** (non fortemente tipizzato)
 - ✓ supporta la programmazione ad oggetti, funzionale e **imperativa**
- ✓ È **Open Source** e ha una vasta *community*

Linguaggi compilati



- ✓ Il programmatore scrive il **codice sorgente** in un file di testo
- ✓ Il **compilatore** traduce il codice sorgente e produce un **codice eseguibile** (linguaggio macchina)
- ✓ L'esecutore carica il codice eseguibile nella memoria del computer e lo esegue
- ✓ Es. C, C++



Linguaggi interpretati



- ✓ Il programmatore scrive il **codice sorgente** in un file di testo
- ✓ L' **interprete traduce** (ad ogni esecuzione) il codice sorgente, trasformandolo in **istruzioni del linguaggio macchina** che vengono eseguite
- ✓ Es. python, php



Why Python?



- ✓ Progettato per essere **semplice** da leggere e scrivere
- ✓ Utilizzato per:
 - ✓ Sviluppo web (Django, Flask)
 - ✓ Data Science e Machine Learning
 - ✓ Automazione, scripting, giochi, applicazioni desktop
- ✓ Ampiamente adottato nella ricerca, nell'industria e nella didattica

Installazione



- ✓ Installazione tramite link ufficiale
<https://www.python.org/doc/versions>

- ✓ tramite installer integrato:

- ✓ Linux: apt get

```
sudo apt update  
sudo apt install python3 python3-pip
```

- ✓ Mac: homebrew

```
brew install python
```

python3 e pip3 vengono installati automaticamente

- ✓ Verifica

```
python --version  
pip --version
```

```
python3 --version  
pip3 --version
```

- ✓ Cos'è pip?

pip è il gestore di pacchetti per Python.

Ti consente di installare librerie aggiuntive come requests, pandas, flask, ecc.

```
pip install nome-pacchetto
```



Strumenti utili





<https://jupyter.org/>

- ✓ Applicazione web interattiva per scrivere ed eseguire codice Python
- ✓ Perfetta per data science, analisi interattiva e prototipazione
- ✓ Ogni cella può contenere codice, output, testi Markdown o immagini
- ✓ Si può installare con **pip** o viene incluso con Anaconda

Google Colab



<https://colab.google/>

- ✓ Versione cloud gratuita di Jupyter Notebook offerta da Google
- ✓ Non richiede installazione né configurazione
- ✓ Supporta l'uso di CPU, GPU e TPU gratuite
- ✓ Ideale per studenti e progetti collaborativi





<https://www.anaconda.com/docs/getting-started/anaconda/install>

- ✓ Distribuzione Python pensata per **Data Science**
- ✓ Include Python, Jupyter Notebook, Spyder, pandas, numpy e altri strumenti scientifici
- ✓ Contiene **conda**, un gestore di pacchetti e ambienti virtuali
- ✓ Ottimo per chi lavora su progetti complessi o in ambito accademico/professionale

Sintassi di base



- ✓ Il codice Python non ha bisogno di tag di apertura/chiusura
- ✓ Le istruzioni non richiedono il punto e virgola (ma è accettato)
- ✓ L'indentazione è obbligatoria e definisce i blocchi di codice
- ✓ I commenti possono essere:
 - ✓ Su una riga usando #
 - ✓ Su più righe: usando triple virgolette "commento " o """commento """

```
age = 17;  
if age < 18:  
    print("Minorenne")  
    print("Non puoi proseguire sul sito")  
else:  
    print("Adulto")
```

```
# questo è un commento su una riga  
...  
questo è un commento  
su più righe  
...  
.....  
anche questo è un commento  
su più righe  
.....
```

Variabili



- ✓ Nome di una variabile (identificatore):
 - ✓ Inizia con una lettera (a-z, A-Z) o un underscore _
 - ✓ Formato da lettere, cifre e underscore ‘_’
 - ✓ Lunghezza illimitata
 - ✓ Non può iniziare con una cifra
 - ✓ Nessun simbolo speciale come \$, -, @, ecc.
 - ✓ Case sensitive: **nome** ≠ **Nome** ≠ **NOME**

- ✓ Una variabile viene creata nel momento in cui viene assegnata la prima volta

```
quantity = 0  
costo = 0.00  
...  
quantity = q
```

Tipi di dato



Tipo	Descrizione	Esempio
int	Numeri interi	x = 5
float	Numeri decimali	pi = 3.14
str	Stringhe di testo	nome = "Ares"
bool	Valori logici: True o False	attivo = True
list	Liste di valori ordinati	numeri = [1, 2, 3]
dict	Dizionari (coppie chiave:valore)	persona = {"nome": "Ada"}
tuple	Sequenze immutabili	coordinate = (4, 5)
set	Insiemi non ordinati senza duplicati	colori = {"rosso", "blu"}

Esistono altri tipi “speciali”:

- ✓ **None**: variabili cui non è assegnato un valore o sono state assegnate con NULL
- ✓ Oggetti/risorse: istanze di classi, connessioni a file o database

Tipizzazione delle variabili in Python



- ✓ Le variabili sono molto “elastiche” nell’assegnamento del tipo di dati

```
quantity = 0  
quantity = 'Hello'
```

- ✓ Casting

```
x = 10  
costo = float(x) # 10.0
```

Tipizzazione debole e liste (I)



- ✓ Le liste sono **dinamiche e eterogenee**, cioè possono contenere elementi di tipo diverso.

Dichiarazione di una lista

```
numeri = [1, 2, 3, 4, 5]
frutti = ["mela", "banana", "ciliegia"]
misto = [1, "due", 3.0, True]
```

Accesso e modifica

```
print(frutti[1]) # banana
frutti[0] = "pera" # sostituisce mela con pera
```

Liste multidimensionali (array 2D)

```
matrice = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
print(matrice[1][2]) # 6
```

Tipizzazione debole e liste (II)



- ✓ Le liste sono **dinamiche e eterogenee**, cioè possono contenere elementi di tipo diverso.

Funzioni e metodi utili

```
len(frutti) # lunghezza  
frutti.append("kiwi") # aggiunge elemento  
frutti.pop() # rimuove ultimo elemento  
"mela" in frutti # verifica presenza
```

Altre tipologie di dati



✓ Dizionari

```
persona = {"nome": "Ada"}  
print(persona["nome"]) # stampa "Ada"  
print(persona.get("nome")) # stampa "Ada"  
persona["cognome"] = "Lovelace" # aggiunge chiave  
print(persona) # stampa {"nome": "Ada", "cognome": "Lovelace"}
```

✓ Coordinate

```
coordinate = (4, 5)  
print(coordinate) # stampa (4, 5)  
print(coordinate[0]) # stampa 4  
coordinate[0] = 10 # errore, le tuple sono immutabili
```

✓ Set

```
colori = {"rosso", "blu"}  
print(colori) # stampa {"rosso", "blu"}  
colori.add("verde") # aggiunge "verde"  
print(colori) # stampa {"rosso", "blu", "verde"}  
print("rosso" in colori) # True  
colori.add("rosso") # non aggiunge duplicato  
print(colori[0]) # errore, gli insiemi non supportano l'indicizzazione  
print(len(colori)) # stampa 3  
colori.pop() # rimuove un elemento casuale  
print(colori['rosso']) # errore, gli insiemi non supportano le chiavi  
for colore in colori:  
    print(colore) # stampa ogni colore nell'insieme
```

Variabili predefinite



Python non dispone di "superglobali" come PHP (es. \$_SERVER), ma fornisce moduli e funzioni per accedere a dati simili

```
import os  
print(os.environ) # stampa le variabili d'ambiente
```

Per accedere a informazioni HTTP in un contesto web, si utilizzano framework come Flask o Django. Esempio con Flask:

```
from flask import request  
  
@app.route("/")  
def index():  
    user_agent = request.headers.get('User-Agent')  
    return f"Il tuo browser è: {user_agent}"
```

Non esiste un equivalente diretto a \$_SERVER["HTTP_HOST"] o \$_SERVER["PHP_SELF"], ma strumenti analoghi sono forniti dai web framework.

Lavorare con le variabili



In Python puoi inserire facilmente variabili all'interno di stringhe:

```
''' Metodo 1: f-string (consigliato) '''
qbanane = 3
print(f"{qbanane} banane")

''' Metodo 2: concatenazione '''
qbanane = 3
print(str(qbanane) + " banane")

''' Metodo 3: format() '''
qbanane = 3
print("{} banane".format(qbanane))

''' Esempi '''
nome = "Ada"
cognome = "Lovelace"

print(f"Ciao {nome}") # stampa Ciao Ada
print('Ciao {nome}') # stampa letteralmente Ciao {nome}
print(f'Ciao {nome}') # stampa Ciao Ada
print("Ciao {} {}".format(nome, cognome)) # stampa "Ciao Ada Lovelace"
print("Ciao {1} {0}".format(nome, cognome)) # stampa "Ciao Lovelace Ada"
print("Ciao {0} {1}".format(nome, cognome)) # stampa "Ciao Ada Lovelace"
```



Costanti



- ✓ In Python **non esistono** costanti vere, ma per convenzione si usano nomi in MAIUSCOLO

```
PI = 3.14159
```

```
MAX_UTENTI = 100
```

- ✓ Non esiste un meccanismo che impedisce la modifica

```
MAX_UTENTI = 200 # possibile, ma sconsigliato!
```

- ✓ Le costanti **non sono vincolate**, ma il maiuscolo segnala al programmatore che non dovrebbe modificarle

Alternative alle Costanti



Enum

- ✓ Utile per gruppi di costanti semantiche
- ✓ È possibile confrontare valori in modo chiaro:
`if oggi == Giorno.LUNEDI`

```
from enum import Enum

class Giorno(Enum):
    LUNEDI = 1
    MARTEDI = 2
    MERCOLEDI = 3

# Uso
oggi = Giorno.LUNEDI
print(oggi) # Giorno.LUNEDI
print(oggi.name) # LUNEDI
print(oggi.value) # 1
```

dataclass

- ✓ `Frozen=True` rende l'istanza immutabile
(come una costante)
- ✓ Utile per impostazioni condivise o valori globali

```
from dataclasses import dataclass

@dataclass(frozen=True)
class Config:
    DB_HOST: str = "localhost"
    DB_PORT: int = 5432
    DEBUG: bool = False

config = Config()
print(config.DB_HOST) # localhost
```



Operatori (I)



Operatori **aritmetici**

Operatore	Significato	Esempio	Risultato
+	Addizione	$5 + 3$	8
-	Sottrazione	$5 - 2$	3
*	Moltiplicazione	$4 * 2$	8
/	Divisione	$10 / 4$	2.5
//	Divisione intera	$10 // 4$	2
%	Modulo (resto)	$10 \% 4$	2
**	Potenza	$2 ** 3$	8

- ✓ Gli operatori funzionano su int e float
- ✓ L'operatore divisione intera (/) restituisce sempre un float

Operatori (II)



Operatori **su stringhe**

- ✓ Operatore +
- ✓ f-string (più leggibile)
- ✓ join() per concatenare una lista di stringhe
- ✓ Non si può concatenare una stringa con un intero direttamente

```
nome = "Ada"
saluto = "Ciao, " + nome + "!"
print(saluto) # Ciao, Ada!
```

```
nome = "Ada"
print(f"Ciao, {nome}!")
```

```
parole = ["Python", "è", "facile"]
print(" ".join(parole)) # Python è facile
```

```
eta = 30
print("Hai " + eta + " anni") # Errore!
print("Hai " + str(eta) + " anni") # Corretto
```

Operatori (III)



Operatore di assegnamento

<variabile> = valore

```
b = 6  
a = 5  
b = 6 + a # stampa 11
```

```
b = 6 + (a = 5) # errore non è possibile assignare a una variabile in  
un'espressione
```

Operatori di assegnamento combinato

Operatore	Esempio	Equivalenti a
<code>+=</code>	<code>x += 1</code>	<code>x = x + 1</code>
<code>-=</code>	<code>x -= 2</code>	<code>x = x - 2</code>
<code>*=</code>	<code>x *= 3</code>	<code>x = x * 3</code>
<code>/=</code>	<code>x /= 4</code>	<code>x = x / 4</code>
<code>//=</code>	<code>x //= 2</code>	<code>x = x // 2</code>
<code>%=</code>	<code>x %= 5</code>	<code>x = x % 5</code>
<code>**=</code>	<code>x **= 2</code>	<code>x = x ** 2</code>

Operatori (IV)



Python **non support** operatore di **incremento** (++) e **decremento** (--)

Operatori di **confronto**

Operatore	Significato	Esempio	Risultato
<code>==</code>	Uguale a	<code>5 == 5</code>	<code>True</code>
<code>!=</code>	Diverso da	<code>5 != 3</code>	<code>True</code>
<code>></code>	Maggiore di	<code>7 > 4</code>	<code>True</code>
<code><</code>	Minore di	<code>2 < 1</code>	<code>False</code>
<code>>=</code>	Maggiore o uguale	<code>5 >= 5</code>	<code>True</code>
<code><=</code>	Minore o uguale	<code>3 <= 2</code>	<code>False</code>

Operatori (V)



Operatore **identità**

Operatore	Significato	Esempio	Risultato
is	Stesso oggetto in memoria (identità)	a is b	True / False
is not	Oggetti diversi	a is not b	True / False

```
a = [1, 2, 3]
b = a
c = [1, 2, 3]

print(a == c) # True → stessi valori
print(a is c) # False → oggetti diversi
print(a is b) # True → stesso oggetto

if variabile is None:
    print("Variabile non inizializzata")
```

Operatori (VI)



Operatori logici

Operatore	Significato	Esempio	Risultato
and	E logico	True and False	False
or	O logico	True or False	True
not	Negazione logica	not True	False

```
eta = 20
studente = True

if eta > 18 and studente:
    print("Studente adulto")

if not studente:
    print("Non sei uno studente")
```

Operatori (VII)



Operatore **condizionale** (ternario)

```
( <condizione> if <valore se vera> else <valore se falsa> )
```

```
# valore_se_vero if condizione else valore_se_falso
eta = 20
messaggio = "Adulto" if eta >= 18 else "Minorenne"

print(messaggio) # Adulto

logged_in = True
print("Accesso consentito" if logged_in else "Accesso negato")
```



Type juggling



- ✓ Alcune conversioni di tipo avvengono automaticamente in base agli operatori utilizzati nelle espressioni
- ✓ La conversione non modifica il tipo degli operandi, che rimangono inalterati
- ✓ Python è più rigido di PHP nel type juggling: non tenta conversioni per operazioni ambigue

```
<?php  
  
$numero = 5;  
$testo = "10 banane";  
$somma = $numero + $testo; // PHP converte  
automaticamente "10 banane" in 10  
echo $somma; // 15
```

```
numero = 5  
testo = "10 banane"  
somma = numero + testo # Errore! non puoi sommare un  
numero e una stringa  
  
# Conversione esplicita  
somma = numero + int(testo.split()[0]) # 5 + 10  
  
print(somma) # 15
```

Funzioni di accesso al tipo



PHP	Python equivalente	Cosa fa
is_boolean(\$a)	isinstance(a, bool)	Verifica se la variabile contiene un valore booleano
is_integer(\$a)	isinstance(a, int)	Verifica se la variabile contiene un numero intero
is_float(\$a) / is_double(\$a)	isinstance(a, float)	Verifica se la variabile contiene un numero reale
is_array(\$a)	isinstance(a, list)	Verifica se la variabile è una lista
is_resource(\$a)	nessun equivalente diretto	In Python si usano oggetti o gestori di contesto
is_null(\$a)	a is None	Verifica se la variabile contiene il valore None
is_numeric(\$a)	a.isnumeric() (solo per str)	Verifica se una stringa è convertibile in numero
gettype(\$a)	type(a).__name__	Restituisce il nome del tipo della variabile
—	callable(a)	Verifica se la variabile è una funzione o può essere chiamata

L'istruzione if



```
if<condizione>:  
    <istruzioni>
```

Il blocco <istruzioni> viene eseguito solo se <condizione> è vera

```
quantita_totale = 0  
  
if quantita_totale == 0:  
    print("Non hai ordinato alcun articolo!")  
  
costo_totale = 32000  
  
if costo_totale >= 30000:  
    print("Hai diritto ad uno sconto del 10%")
```



L'istruzione else



```
else:  
    <istruzioni>
```

Il blocco <istruzioni> viene eseguito solo se la condizione del precedente **if** è falsa

```
qpomodori = 2  
qbanane = 3  
qmele = 4  
quantita_totale = qpomodori + qbanane + qmele  
  
if quantita_totale == 0:  
    print("Non hai ordinato alcun articolo!<br />")  
else:  
    print("Ecco la lista degli articoli:<br />")  
    print("<ul>")  
    print(f"<li>{qpomodori} pomodori</li>")  
    print(f"<li>{qbanane} banane</li>")  
    print(f"<li>{qmele} mele</li>")  
    print("</ul>")
```

L'istruzione elseif



È usato quando si hanno più di due rami decisionali:

```
elif <condizione>:  
<istruzioni>
```

```
if costo_totale <= 10000:  
    sconto = 0.0  
elif costo_totale <= 20000:  
    sconto = 0.10  
elif costo_totale <= 40000:  
    sconto = 0.20  
elif costo_totale <= 80000:  
    sconto = 0.25  
else:  
    sconto = 0.30
```

Viene eseguito solo il blocco di istruzioni corrispondente alla prima condizione vera

Se nessuna condizione vera, viene eseguito il blocco della **else** (se specificata)

L'istruzione switch (I)



In Python non esiste l'istruzione **switch** come in PHP, C o JavaScript

```
switch ($nome)
{
    case 'Luca':
    case 'Giorgio':
    case 'Franco':
        echo "Ciao, vecchio amico!"; break;
    case 'Mario': echo "Ciao, Mario!"; break;
    default: print "Benvenuto, chiunque tu sia";
}
```

Uso di if-elif-else

```
nome = "Luca"
if nome in ("Luca", "Giorgio", "Franco"):
    print("Ciao, vecchio amico!")
elif nome == "Mario":
    print("Ciao, Mario!")
else:
    print("Benvenuto, chiunque tu sia")
```

L'istruzione switch (II)



Con dizionario di risposte

```
nome = "Luca"

messaggi = {
    "Luca": "Ciao Luca, vecchio amico!",
    "Giorgio": "Ciao Giorgio, vecchio amico!",
    "Franco": "Ciao Franco, vecchio amico!",
    "Mario": "Ciao, Mario!"
}

print(messaggi.get(nome, "Benvenuto, chiunque tu sia"))
```

Python 3.10+ – Con match (pattern matching)

```
match nome:
    case "Giorgio" | "Franco":
        print("Ciao, vecchio amico!")
    case "Luca":
        print("Ciao, Luca!")
    case "Mario":
        print("Ciao, Mario!")
    case _:
        print("Benvenuto, chiunque tu sia")
```



Ciclo while



```
while <condizione>:  
    <istruzioni>
```

Il blocco <istruzioni> viene eseguito fino a quando <condizione> è vera

```
x = 0  
while x < 5:  
    print(x)  
    x += 1
```

In Python tutti i cicli condizionati si fanno con **while**, **non esiste** il **do...while**

Ciclo for



```
for variable in range(start, stop, step):  
    <istruzioni>
```

```
for i in range(5):  
    print(i) # stampa 0, 1, 2, 3, 4  
  
for i in range(1, 10, 2):  
    print(i) # stampa 1, 3, 5, 7, 9  
  
nomi = ["Luca", "Sara", "Giorgio"]  
for nome in nomi:  
    print(f"Ciao, {nome}")
```



Funzioni



```
def nome_funzione(parametri):
    # corpo della funzione
    valore = 5
    return valore
```

Argomenti variabili: *args e **kwargs

```
def somma(*numeri):
    return sum(numeri)

print(somma(1, 2, 3)) # 6
```

```
def stampa_info(**info):
    for chiave, valore in info.items():
        print(f'{chiave}: {valore}')

stampa_info(nome="Luca", eta=30)
```



Funzioni anonime - lambda



```
quadrato = lambda x: x * x
print(quadrato(4)) # 16

numeri = [1, 2, 3, 4]
doppio = list(map(lambda x: x * 2, numeri))
print(doppio) # [2, 4, 6, 8]

numeri = [1, 2, 3, 4, 5, 6]
pari = list(filter(lambda x: x % 2 == 0, numeri))
print(par) # [2, 4, 6]

parole = ["banana", "kiwi", "mela"]
ordinate = sorted(parole, key=lambda x: len(x))
print(ordinate) # ['kiwi', 'mela', 'banana']
```



PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - PYTHON





PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - DJANGO

Prof. Ada Bagozi

ada.bagozi@unibs.it



Cos'è il Django?



- ✓ *Django è un framework web open source scritto in Python*
- ✓ *Favorisce uno sviluppo rapido e sicuro*
- ✓ *Filosofia Batteries included – molte funzionalità integrate*



django

P
W

Caratteristiche principali



- ✓ *ORM integrato*
semplifica l'interazione con i database, permettendo agli sviluppatori di interagire con le tabelle del database usando oggetti Python invece di SQL
- ✓ *Sistema di template potente (sintassi { % ... % })*
- ✓ *Interfaccia di amministrazione automatica*
- ✓ *Sicurezza integrata (CSRF, XSS, SQL injection)*
- ✓ *Supporto per API REST (Django REST Framework)*

Architettura MVC (Laravel)



- ✓ **Model**: definisce la struttura dei dati, ovvero quali dati l'app debba contenere
- ✓ **View**: descrive come i dati dell'app vadano mostrati
- ✓ **Controller**: contiene la logica che aggiorna i Model e/o le View in risposta agli input forniti dall'utente



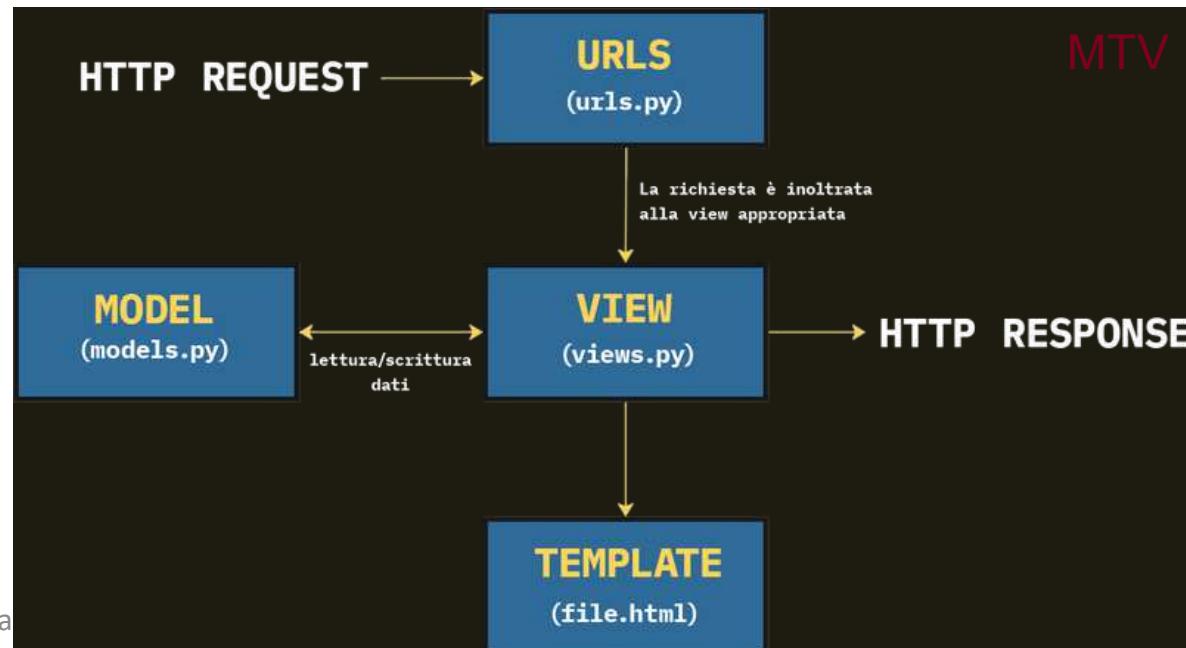
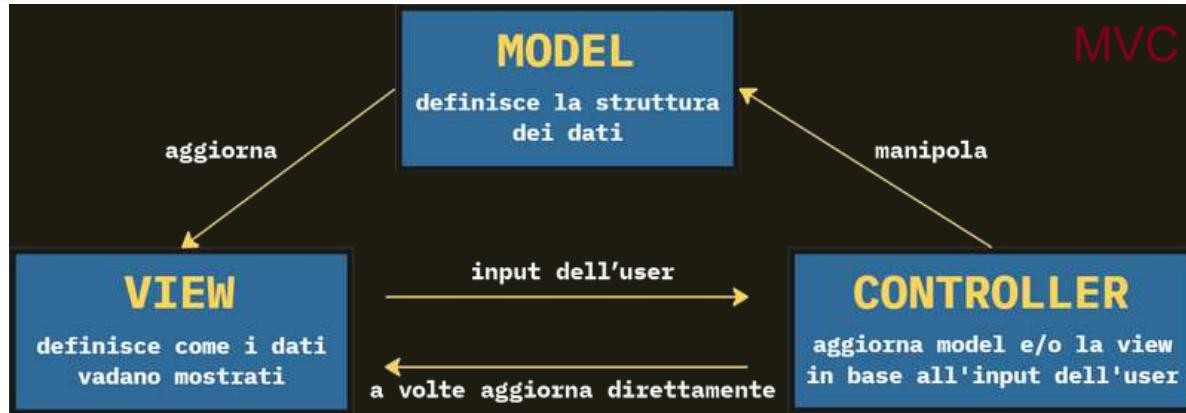
<https://www.programmareinpython.it/blog/che-cosa-rende-django-speciale-miglior-web-framewo/>

Architettura MTV (Django)



- ✓ **Model:** Definisce la struttura delle entità del nostro sito, tradotte poi in tabelle nel database (data access layer)
- ✓ **Template:** Descrive come i dati vadano mostrati nelle pagine web (presentation layer)
- ✓ **View:** Gestisce le richieste e le risposte HTTP, dispone della logica per sapere a quali dati accedere tramite i model e delega la formattazione della risposta ai template (business logic layer)

MCV vs MTV



Django vs. Laravel (I)



Criterio	Django (Python)	Laravel (PHP)
Linguaggio	Python – leggibile, usato anche in data science	PHP – molto diffuso nel web, facile da trovare hosting
Architettura	MTV (Model–Template–View) semplice e diretto	MVC (Model–View–Controller) ben noto e separazione netta
ORM	ORM integrato, potente e semplice da usare	Eloquent ORM, intuitivo e fluente
Sicurezza	Protezioni predefinite (CSRF, XSS, SQLi, clickjacking)	Protezioni buone, ma più configurabili manualmente
Admin	Interfaccia admin automatica out-of-the-box	Pacchetti come Laravel Nova (più personalizzabili)
CLI	manage.py – semplice, comandi ben strutturati	artisan – molto potente e ricco di comandi utili
Routing	Esplicito e leggibile	Elegante e supporta route closures
Template Engine	Django Template Language – chiaro e sicuro	Blade – flessibile, supporta estensioni e direttive personalizzate
APIs REST	Django REST Framework – completo e robusto	Laravel Sanctum, Passport – moderne soluzioni per API



Django vs. Laravel (II)



Criterio	Django (Python)	Laravel (PHP)
Performance	Efficiente, adatto a progetti complessi	PHP 8 ha migliorato molto le prestazioni
Ecosistema	Ottima integrazione con strumenti Python (es. AI, ML, scientific computing)	Ricco ecosistema di pacchetti Composer
Community	Attiva nel mondo Python, scientifico e accademico	Vibrante, tutorial/video abbondanti, grande supporto PHP
Hosting	Può richiedere configurazione su server (es. WSGI, ASGI)	Ampio supporto su hosting tradizionali (shared hosting, cPanel, ecc.)
Documentazione	Ufficiale molto dettagliata, chiara e autorevole	Chiara, con molti esempi, screencast e risorse community
Curva di apprendimento	Richiede familiarità con Python e il pattern MTV	Rapida per chi già conosce PHP



Installazione di Django



Install django and create project

```
pip install django  
django-admin startproject my_project  
cd project  
python manage.py runserver
```

Create app

```
django-admin startapp my_app
```

```
my_project/  
└── manage.py  
└── my_project/  
    ├── __init__.py  
    ├── asgi.py  
    ├── settings.py  
    ├── urls.py  
    └── wsgi.py  
└── my_app/  
    ├── migrations/  
    │   └── __init__.py  
    ├── __init__.py  
    ├── admin.py  
    ├── apps.py  
    ├── models.py  
    ├── tests.py  
    ├── urls.py  
    └── views.py
```



Directory principale del progetto



- ✓ È il nucleo dell'intero progetto Django
- ✓ Contiene i file di configurazione globali
- ✓ Organizza e coordina le funzionalità dell'intero sistema web

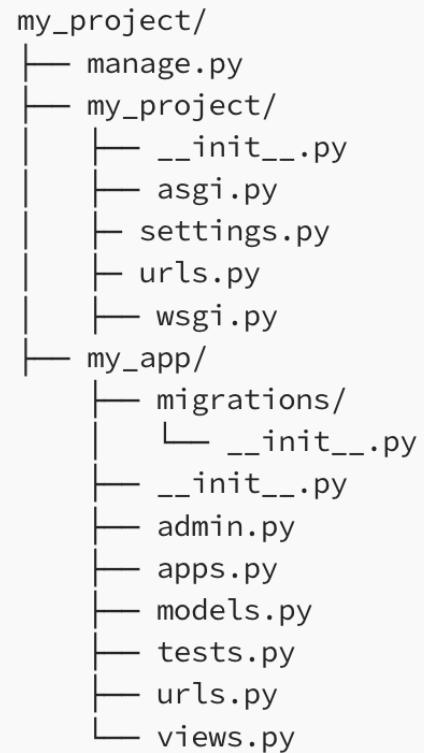
```
my_project/
├── manage.py
└── my_project/
    ├── __init__.py
    ├── asgi.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
└── my_app/
    ├── migrations/
    │   └── __init__.py
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    ├── urls.py
    └── views.py
```

File principali nella directory del progetto



```
django-admin startapp my_app
```

- ✓ manage.py: Interfaccia per i comandi Django (runserver, migrate, etc.)
- ✓ settings.py: Configurazione generale (database, app, middleware)
- ✓ urls.py: Dispatcher delle URL: collega URL a viste
- ✓ wsgi.py: Gateway per server di produzione (WSGI)
- ✓ asgi.py: Gateway asincrono per ASGI server
- ✓ init.py: Rende la directory un package Python



Directory di un'app Django



- ✓ Ogni app è un modulo indipendente all'interno del progetto
- ✓ Struttura generata automaticamente da startapp

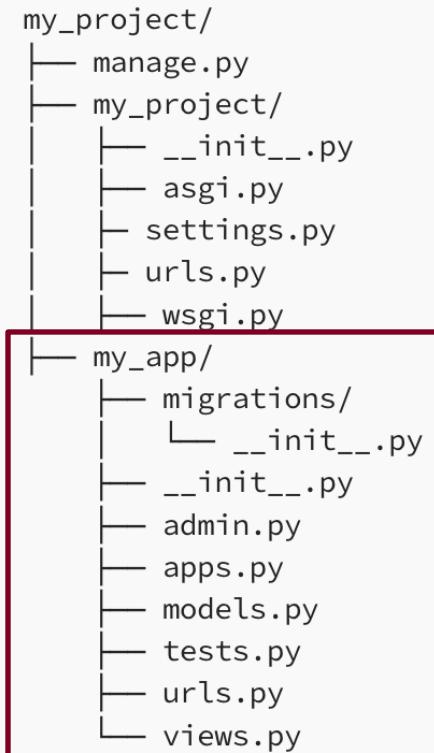
```
my_project/
├── manage.py
└── my_project/
    ├── __init__.py
    ├── asgi.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
└── my_app/
    ├── migrations/
    │   └── __init__.py
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    ├── urls.py
    └── views.py
```



Directory di un'app Django



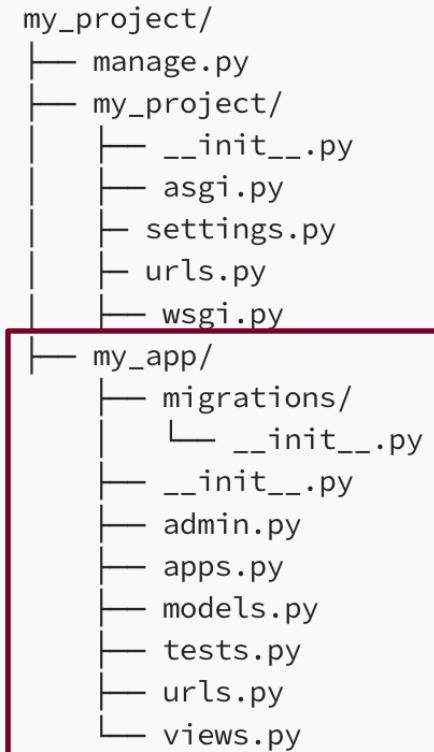
- ✓ `models.py`: Definizione dei dati tramite ORM
- ✓ `views.py`: Logica delle richieste/risposte
- ✓ `admin.py`: Configurazione dell'interfaccia admin
- ✓ `tests.py`: Test automatici
- ✓ `migrations/`: Tracciamento delle modifiche ai modelli
- ✓ `apps.py`: Configurazione dell'app (Opzionale)
- ✓ `forms.py`, `urls.py`, `utils.py`



Il concetto di App Riutilizzabili



- ✓ Ogni app può essere:
 - ✓ Indipendente
 - ✓ Trasferibile tra progetti
 - ✓ Manutenibile separatamente
- ✓ **Vantaggi:**
 - ✓ Modularità e chiarezza
 - ✓ Codice riutilizzabile
 - ✓ Standardizzazione tra progetti
 - ✓ Facilità di aggiornamento
 - ✓ Contributi dalla community



Esempio di gerarchia di progetto Django



- ✓ static/: file statici (CSS, JS, immagini)
- ✓ media/: contenuti caricati dagli utenti
- ✓ templates/: template HTML condivisi

```
myproject/
├── manage.py
└── myproject/
    ├── __init__.py
    ├── asgi.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
└── blog/
└── utenti/
└── static/
└── media/
└── templates/
```

Convenzioni di nomi



- ✓ App: minuscolo con underscore (es. my_app)
- ✓ File Python: minuscolo con underscore (es. views.py)
- ✓ Classi: CamelCase (es. MyModel)
- ✓ Variabili e funzioni: snake_case (es. get_data)

Buone pratiche di progettazione



- ✓ Separare responsabilità per app
- ✓ Usare Class-Based Views dove utile
- ✓ Inheritance nei modelli
- ✓ Inheritance nei template
- ✓ Modulo utils.py per funzioni comuni

Gestione delle impostazioni (settings.py)



- ✓ Utilizzare variabili di ambiente per credenziali e chiavi
- ✓ Separare settings per ambiente (sviluppo, produzione)
- ✓ Tenere DEBUG = False in produzione

Middleware in Django



- ✓ Il **middleware** è una serie di hook che processano la richiesta e la risposta
- ✓ Può essere usato per:
 - ✓ Autenticazione
 - ✓ Compressione delle risposte
 - ✓ Gestione della sessione
 - ✓ Protezione CSRF

```
MIDDLEWARE = [  
    "django.middleware.security.SecurityMiddleware",  
    "django.contrib.sessions.middleware.SessionMiddleware",  
    "django.middleware.common.CommonMiddleware",  
    "django.middleware.csrf.CsrfViewMiddleware",  
    "django.contrib.auth.middleware.AuthenticationMiddleware",  
    "django.contrib.messages.middleware.MessageMiddleware",  
    "django.middleware.clickjacking.XFrameOptionsMiddleware",  
]
```

Esempio di Middleware



Un middleware personalizzato per loggare ogni richiesta:

```
class LogMiddleware:  
    def __init__(self, get_response):  
        self.get_response = get_response  
  
    def __call__(self, request):  
        print(f"Request path: {request.path}")  
        response = self.get_response(request)  
        return response
```

Aggiungerlo in settings.py:

```
MIDDLEWARE = [  
    'mia_app.middleware.LogMiddleware',  
    ...  
]
```



Controllo degli Accessi alle Viste



Usare il decoratore @login_required

```
from django.contrib.auth.decorators import login_required

@login_required
def area_riservata(request):
    return render(request, 'riservata.html')
```

Per le class-based views

```
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import TemplateView

class AreaRiservataView(LoginRequiredMixin, TemplateView):
    template_name = 'riservata.html'
```

Se l'utente non è autenticato, verrà reindirizzato alla login.



Modelli in Django (models.py)



- ✓ Ogni modello è una classe Python che rappresenta una tabella nel database
- ✓ Utilizza il sistema ORM integrato per creare/modificare dati

```
from django.db import models

class Author(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

```
CREATE TABLE myapp_author (
    "id" bigint NOT NULL PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

Le modifiche ai modelli si riflettono nel database tramite `makemigrations` e `migrate`

<https://docs.djangoproject.com/en/5.2/topics/db/models/>



Configurazione del Database



- ✓ In settings.py, sezione DATABASES

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
    }
}
```

- ✓ Altri motori disponibili:
 - ✓ PostgreSQL: 'django.db.backends.postgresql'
 - ✓ MySQL: 'django.db.backends.mysql'
 - ✓ Oracle: 'django.db.backends.oracle'

Modello Book



```
class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.ForeignKey(Author, on_delete=models.CASCADE)
    publication_date = models.DateField()
    isbn = models.CharField(max_length=13, unique=True)
    pages = models.IntegerField()
    cover_image = models.ImageField(upload_to='covers/')
    language = models.CharField(max_length=30)
    summary = models.TextField()
    genre = models.CharField(max_length=30)
```



Esempi di accesso ai dati



```
#Creare un oggetto:  
autore = Author.objects.create(first_name="Umberto", last_name="Eco")  
  
#Recuperare tutti i libri:  
libri = Book.objects.all()  
  
#Recuperare uno specifico libro  
libro = Book.objects.get(id=1)  
  
#Filtrare  
libro = Book.objects.filter(title="Il nome della rosa").first()  
  
#Ordinare  
libri = Book.objects.order_by('titolo')  
#Eliminare  
libro = Book.objects.get(id=1)  
libro.delete()
```



Versionamento del codice e sicurezza



- ✓ Usare Git per versionare
- ✓ Includere un .gitignore
- ✓ Evitare di tracciare:
 - ✓ file .pyc
 - ✓ cartelle __pycache__
 - ✓ cartelle env/, .venv/
 - ✓ db.sqlite3, se locale
 - ✓ ...





PROGRAMMAZIONE WEB

SERVER-SIDE SCRIPTING - DJANGO

