PROGRAMMAZIONE WEB
HTTP E HTML

Prof. Ada Bagozi
ada.bagozi@unibs.it

Architetture client-server

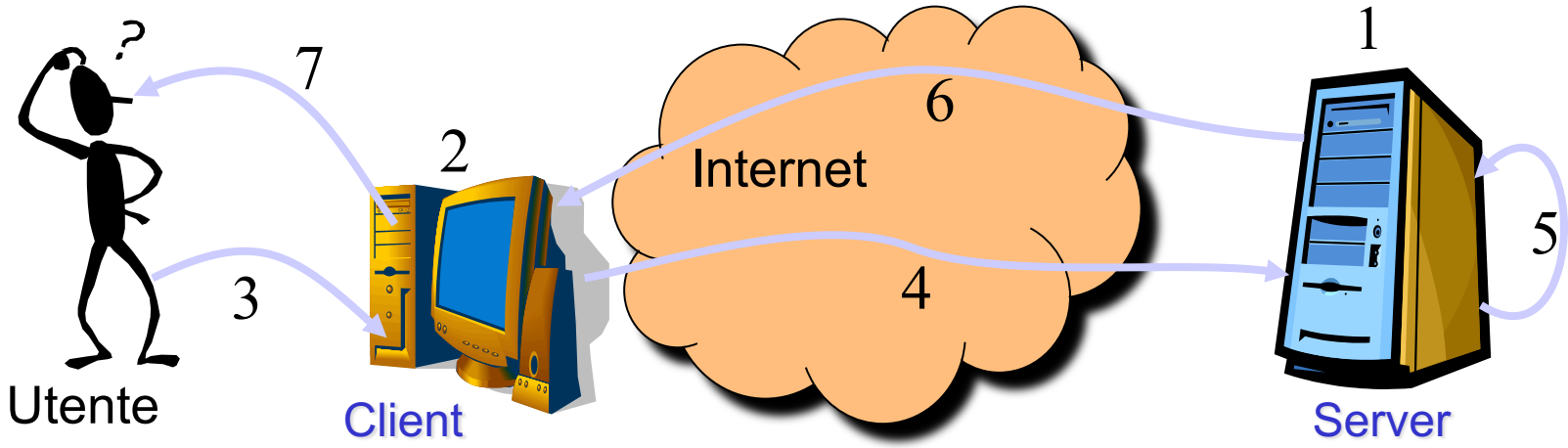
Le applicazioni Web sono organizzate secondo la cosiddetta *architettura client-server*:

✓ 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

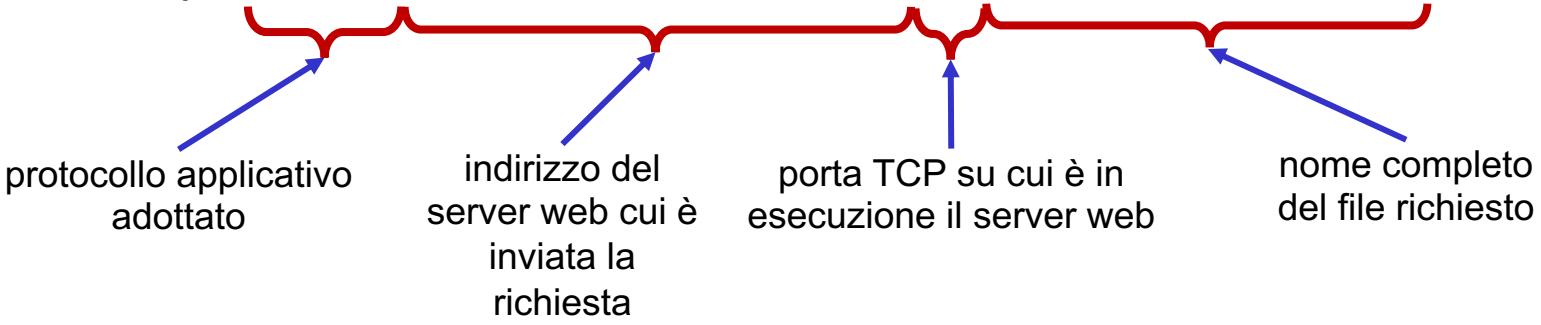
Identificazione delle risorse

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, tabelle, 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>, , , <dfn>,
<code>, <samp>, <kbd>, <var>, <cite>, <abbr>,
**<acronym>, <a>, <img>, <object>,
, <script>,**
<map>, <q>, <sub>, <sup>, , <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>, ,
<noscript>, <blockquote>, <form>, <hr>, <table>,
<fieldset>, <address>

Classificazione dei tag HTML (II)

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

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

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

Tag per la rappresentazione di liste: `<ul>`, `<ol>`, `<li>`, `<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: `<img>`, `<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 tabelle (I)

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

Tag HTML – Le tabelle (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

		Preferenza		
		Pianura	**Collina**	**Montagna**
Categoria	**Maschi**	54%	24%	22%
	Femmine	63%	21%	16%
Dati ottenuti mediante campionamento pseudo-casuale				

Tag HTML – Le tabelle (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

`<img>`: 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 HTTT 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

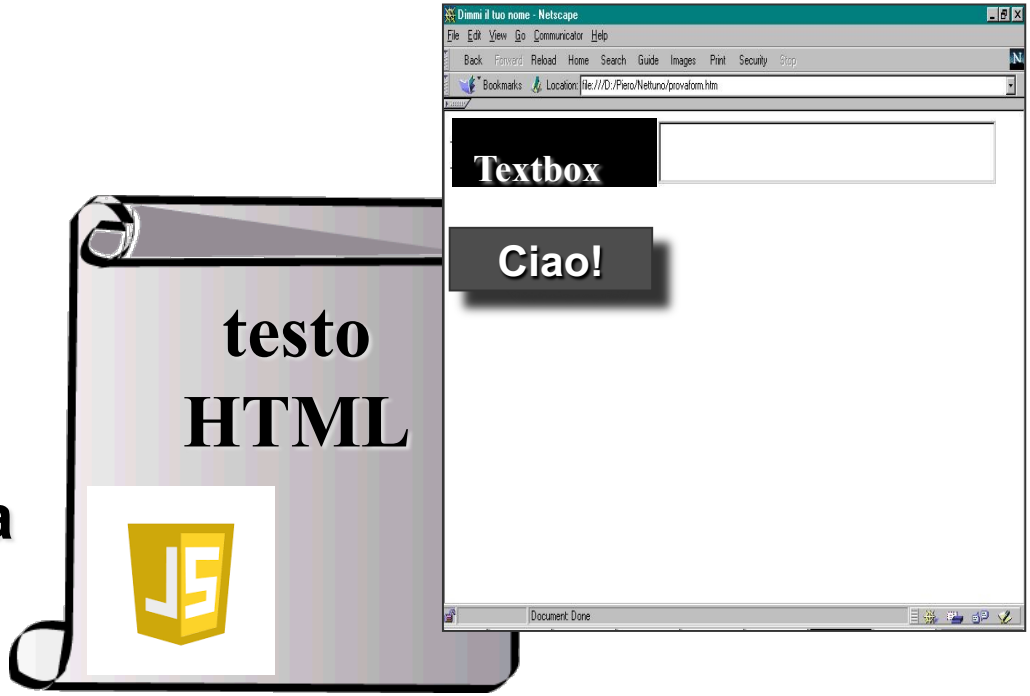
Contenuto

+

script

=

pagina interattiva

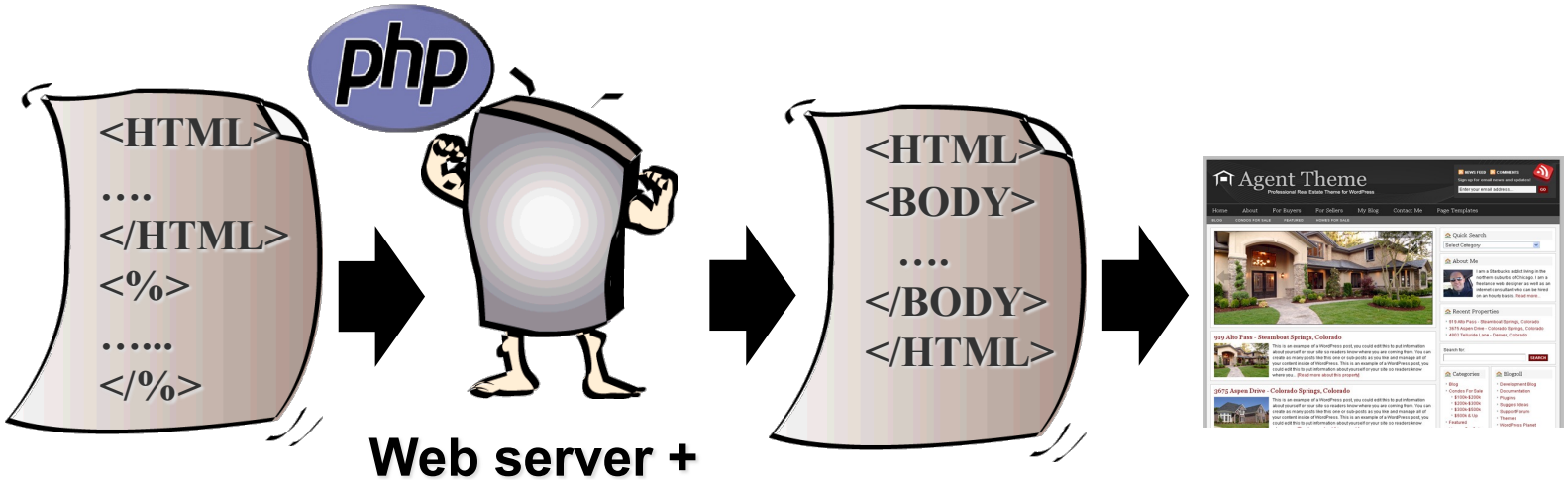


Server-side scripting

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



Web server +
Esecutore script

Architetture multi-tier (I)

Combinazione di due sottosistemi *client-server*:

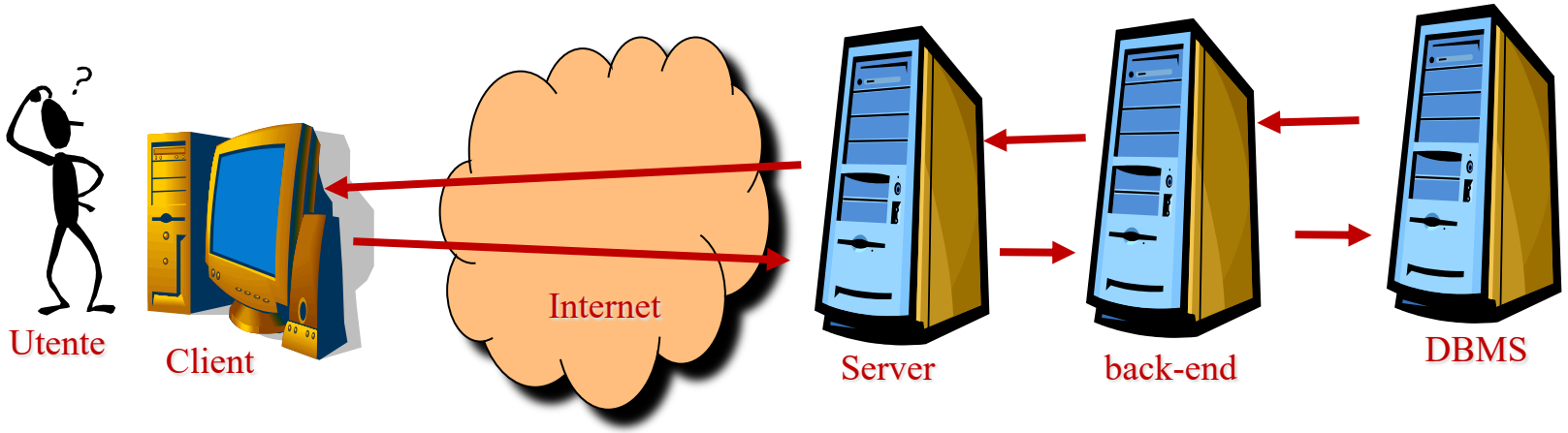
✓ il browser opera come *client* verso il *server web*, inviandogli una richiesta e aspettando da esso una risposta

✓ 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/)
- ✓ E molto altro ancora in rete

PROGRAMMAZIONE WEB
HTTP E HTML