

# *CORSO DI HTML*

*Prof. Fulvio Ferroni*

# Sommario

<b>1</b>	<b><u>INTERNET</u></b>	<b>3</b>
1.1	<u>Cenni storici</u>	3
1.2	<u>I protocolli</u>	4
1.3	<u>INTRANET e EXTRANET</u>	5
1.4	<u>IL WORLD WIDE WEB</u>	6
1.5	<u>Gli indirizzi delle risorse di Internet</u>	10
1.6	<u>Collegarsi a Internet da casa</u>	13
<b>2</b>	<b><u>IL LINGUAGGIO HTML</u></b>	<b>14</b>
2.1	<u>Introduzione</u>	14
2.2	<u>Storia dell'HTML</u>	14
2.3	<u>Aspetto di un file HTML</u>	15
2.4	<u>Strumenti per la creazione di documenti HTML</u>	15
2.5	<u>Struttura di un documento HTML</u>	16
2.6	<u>Nidificazione dei tag</u>	19
2.7	<u>Set di caratteri</u>	19
2.8	<u>Altri elementi di uso frequente</u>	21
2.9	<u>Elenchi o Liste</u>	23
2.10	<u>Collegamenti</u>	25
2.11	<u>Immagini nei documenti</u>	29
<b>3</b>	<b><u>DOCUMENTI AVANZATI</u></b>	<b>31</b>
3.1	<u>Informazioni supplementari nell'intestazione</u>	31
3.2	<u>Uso delle tabelle</u>	32
3.3	<u>Cenni a FRAME, CSS, Oggetti multimediali</u>	39
<b>4</b>	<b><u>PAGINE INTERATTIVE E DINAMICHE</u></b>	<b>48</b>
4.1	<u>Introduzione</u>	48
4.2	<u>Moduli e programmi CGI</u>	48
4.3	<u>Definizione e utilizzo dei programmi CGI</u>	57
4.4	<u>WEB e database, PHP</u>	66
4.5	<u>Script attivi, DHTML, applet, servlet</u>	77
4.6	<u>WEB ad oggetti</u>	85
4.7	<u>Conclusioni</u>	85
<b>5</b>	<b><u>DA HTML A XML: INTRODUZIONE</u></b>	<b>87</b>
5.1	<u>Limiti dell'HTML</u>	87
5.2	<u>Genesi e natura dell'XML</u>	88
5.3	<u>Documenti XML ben formati e validi</u>	88
5.4	<u>Parser ed applicazioni XML</u>	91
<b>6</b>	<b><u>BIBLIOGRAFIA</u></b>	<b>94</b>



# 1 INTERNET

## 1.1 Cenni storici

Internet è una rete internazionale formata dall'interconnessione di molte migliaia di reti di computer. La sua storia inizia negli anni sessanta e precisamente nel 1962 quando negli Stati Uniti viene creata, nell'ambito del Dipartimento della Difesa, un'agenzia di nome DARPA (*Defence Advanced Research Projects Agency*).

Il suo scopo fondamentale era quello di riacquisire il primato tecnologico nei confronti dell'Unione Sovietica che nel 1957 aveva lanciato il primo satellite artificiale della storia, lo Sputnik. L'ARPA doveva in particolare progettare un sistema di telecomunicazioni in grado di funzionare in caso di attacco bellico anche di tipo nucleare. I primi studi portarono alla definizione di quella che sarebbe diventata la rete Arpanet che può essere considerata l'antenata di Internet. Essa non aveva nessuna autorità centrale, i nodi che ne facevano parte erano autonomi ed in grado di operare in una situazione di forte instabilità in modo che la rete potesse sopportare anche la distruzione di uno o più dei nodi stessi.

Il 30 agosto 1969 venne installato il primo nodo presso l'Università della California a Los Angeles dotato di un computer "Processore di messaggi di interfaccia Honeywell numero 1". Nel giro di 3 mesi i nodi divennero quattro: Stanford Research Institute, Università della California a Santa Barbara, Università dello Utah.

Nel corso degli anni '70 molte istituzioni collegarono le loro reti o i loro computer ad Arpanet e gli utenti iniziarono ad usare la rete principalmente per scambiarsi a distanza messaggi sotto forma di posta elettronica o per inviare e ricevere file contenenti dati o programmi.

All'inizio degli anni '80, la rete cominciò ad espandersi in modo massiccio divenendo a tutti gli effetti una "rete di reti" e utilizzando Arpanet come dorsale (rete ad alta velocità che unisce tra loro altre reti locali). Nel 1981 i computer collegati erano 213.

Rimanevano però esclusi gli atenei e i centri di ricerca che non avevano rapporti con il Dipartimento della Difesa degli Stati Uniti. Per potere estendere l'accesso a tutti gli interessati fu necessario il disimpegno dei militari che nel 1983 crearono una loro rete (Milnet) mentre Arpanet assunse il nome e le caratteristiche di Internet. Nel 1985 vi si collegavano già 100 reti diverse, salite poi a 500 nel 1989, saturando completamente la capacità della dorsale Arpanet. Fu a questo punto che la NSF (*National Science Foundation*), istituita dal governo americano con lo scopo di favorire la crescita di sistemi di comunicazione tra le università, decise di creare la nuova dorsale Nfsnet al fine di sostituire Arpanet che fu infatti definitivamente smantellata nel 1990.

Adesso esistono molte altre dorsali affiancate a Nfsnet ed è così possibile il collegamento ad Internet di migliaia di reti (oltre 50.000 nel 1995) e milioni di singoli computer o *HOST* (5 nel 1995 saliti a 36 a fine 1998) attraverso vari mezzi trasmissivi come: cavi in fibra ottica, cavi coassiali, cavi telefonici, satelliti, onde radio.

Nella tabella seguente viene riepilogato il numero di utenti di Internet nel Mondo, negli USA e in Italia negli ultimi anni (i dati sono espressi in milioni):

Anno	Mondo	USA	Italia
1995	14	10	0,45
1996	38	23	
1997	90	40	2,3
1998	142	63	
1999	200	80	5
2000	260	105	10



Come si vede da queste cifre in Italia il numero di utenti è stato a lungo molto basso rispetto al totale ma ultimamente sta crescendo con grande rapidità. Quindi anche nel nostro paese Internet si sta sempre più affermando come fondamentale mezzo di comunicazione e anche come uno strumento di lavoro efficace e uno sbocco commerciale di notevole importanza; molte aziende infatti stanno spostando del tutto o in parte la loro attività pubblicitaria e commerciale sulla rete dando impulso al cosiddetto *E-commerce* (*commercio elettronico*).

La caratteristica più "bella" e interessante di Internet è che nessuno ne è "proprietario"; tutti gli enti che sono collegati alla rete ne gestiscono una parte e sono responsabili di una frazione dell'immensa mole di informazioni in essa contenute. In ultima analisi anche un utente finale (cioè chiunque di noi) che crea e pubblica delle pagine e le diffonde in rete diviene "comproprietario" di Internet.

Siamo dunque in presenza di un mezzo di comunicazione moderno, in continua espansione, diffuso in tutto il Mondo e soprattutto libero. La differenza con le televisioni, i giornali, le case editrici, discografiche e cinematografiche è infatti evidente: esse sono sempre o quasi di proprietà di qualcuno che, volendo, può manipolarle a piacimento in modo da influenzare le opinioni, i gusti e le scelte degli "utenti" per i propri scopi più o meno leciti. Inoltre con tali mezzi difficilmente si riesce ad avere un ruolo attivo nel processo di comunicazione, cosa invece spesso possibile con Internet.

Grazie al computer e alle reti è nato dunque questo strumento di comunicazione e di condivisione delle informazioni totalmente libero con ciò smentendo le paure di molte persone (esprese anche in tanti libri o film) che temevano che il calcolatore potesse divenire una sorta di "Grande Fratello" in grado di controllare e condizionare la vita di tutti i cittadini limitandone in modo inaccettabile la libertà. Almeno per ora queste macchine così bistrattate sono servite esattamente al contrario: sono infatti uno strumento di libertà di cui praticamente tutti possono usufruire (almeno nei paesi economicamente avanzati).

Visto che siamo entrati in questi ragionamenti "filosofici", può essere opportuno aggiungere qualche parola su certe "campagne" giornalistiche che, basandosi sulla sostanziale ignoranza (nel senso di non conoscenza) del fenomeno, sia da parte del grande pubblico sia, cosa molto più grave, da parte dei giornalisti che le conducono, tendono a criminalizzare Internet dipingendola come la fonte delle peggiori nefandezze ed arrivando ad affermare che darebbe addirittura assuefazione come la droga. Questo modo di dipingere la situazione non fa altro che portare acqua al mulino di chi, a vari livelli, vorrebbe introdurre vincoli, limitazioni e, alla fine, censure all'interno della rete e riguardo al suo utilizzo.

Nessuno può negare che Internet venga usata anche per scopi illeciti, ma questo avviene in una percentuale irrisoria di casi e comunque non si dovrebbe incolpare lo strumento per l'uso che ne viene fatto: se un coltello viene usato per ferire una persona invece che per affettare il prosciutto è colpa del coltello? O ancora: si deve proibire l'uso dei telefonini visto che sono molto utili ai personaggi mafiosi per mantenere i loro collegamenti?

Morale: siamo solo in presenza di uno strumento del quale si può fare un uso buono o cattivo ma, in ogni caso, la responsabilità è di chi lo usa e non dello strumento.

## 1.2 I protocolli

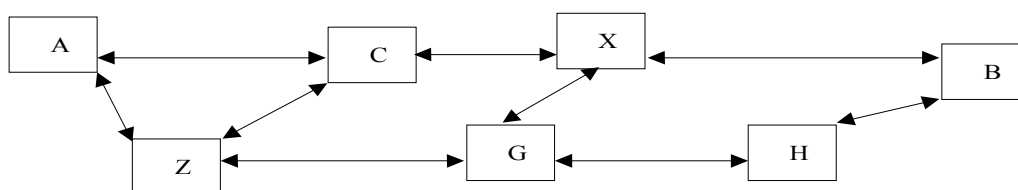
Il funzionamento di Internet è basato su tutta una serie di *protocolli*. Per protocollo si intendono la procedura e le regole da seguire nel trasmettere e ricevere dati su una rete oppure su una linea di comunicazione in modo che gli utenti possano usare macchine elettroniche per scambiare informazioni in modo ordinato così da riprodurre fedelmente al punto di ricezione quello che è stato inviato dal punto di trasmissione. Il protocollo specifica il formato dei dati, la temporizzazione dei segnali, la sequenza d'invio e i sistemi per verificare la presenza di eventuali errori di trasmissione.

Prima di parlare dei protocolli di Internet è opportuno introdurre un concetto molto importante su cui essi sono basati, quello della differenza tra *client* e *server*. Un server è un computer (con apposito software) che fornisce informazioni o servizi; un client è uno strumento o programma, di solito su un computer diverso, che usufruisce delle informazioni o dei servizi.

La caratteristica fondamentale di tutti i protocolli di Internet è il loro essere gratuiti e "aperti" cioè in grado di funzionare con la quasi totalità delle reti fisiche e in maniera indipendente dall'hardware di un particolare produttore.



Il protocollo più importante è senza dubbio il TCP/IP (*Transfer Control Protocol / Internet Protocol*) basato sulla tecnologia a commutazione di pacchetto. I dati da inviare vengono suddivisi nel nodo (computer) di partenza in diverse parti (*packet*) il cui cammino attraverso la rete può seguire differenti percorsi; nel nodo di arrivo vengono poi ricomposti controllandone anche l'integrità. Un primo vantaggio di tale metodo consiste nel fatto che non è necessario né definire né conoscere il cammino da compiere; è il software che, lungo la rete, si preoccupa di instradare i dati evitando le eventuali interruzioni e scegliendo, comunque, il percorso più veloce. Un altro vantaggio è che vengono sfruttati al meglio i canali trasmissivi che non sono mai interamente occupati da un'unica trasmissione (come avviene ad esempio nelle comunicazioni telefoniche nelle quali la "linea" tra il chiamante ed il ricevente è occupata per tutta la durata della telefonata). Quindi se ipotizziamo che il computer A invii dati al computer B, le due macchine non devono necessariamente essere collegate in modo diretto; saranno presenti altri "nodi" intermedi attraverso i quali verranno smistati i pacchetti che, allo scopo, contengono al loro interno l'indirizzo del nodo di destinazione (oltre che di quello di origine).



Nell'esempio precedente se ipotizziamo che il messaggio inviato da A a B sia suddiviso in tre pacchetti potremmo avere i seguenti percorsi:

pacchetto 1: A - Z - G - X - B

pacchetto 2: A - C - X - B

pacchetto 3: A - Z - G - H - B

Approfondimenti sul protocollo TCP/IP e sulle reti di computer in generale esulano dagli scopi di questo corso, si consiglia la consultazione di testi specifici.

Altri protocolli e servizi importanti in Internet sono:

- TELNET per collegarsi a distanza ad un computer;
- SMTP e POP3 per la posta elettronica (*E-mail*);
- FTP per lo scambio di file;
- NNTP per la distribuzione di messaggi nei *newsgroup* (bacheche elettroniche);
- HTTP per il trasferimento di documenti ipertestuali ("pagine" di informazioni presenti in rete).

## 1.3 INTRANET e EXTRANET

Una Intranet è una rete invariabilmente imperniata sul protocollo TCP/IP e che spesso utilizza *server WEB* (vedremo più avanti cosa è) come punto di centralizzazione e pubblicazione delle informazioni. Viene realizzata entro i confini di una azienda o di una qualunque organizzazione (anche una scuola) ed è invisibile o solo parzialmente visibile dall'esterno o perché non è collegata con Internet o perché è collegata tramite un computer *firewall* che ha il compito di proteggere i dati dalle intrusioni esterne non desiderate. Viene utilizzata come struttura per convogliare informazioni utili alle attività interne e per facilitarne l'uso da parte



degli utenti. In una Intranet si usano infatti gli stessi strumenti di Internet, come ad esempio i browser grafici che sono molto "amichevoli" e facili da usare e i programmi per la gestione della posta elettronica.

Una Extranet è invece l'opposto di Internet, nel senso che è una rete geografica di ampia portata sempre imperniata sul protocollo TCP/IP che però viene utilizzata unicamente per scopi privati e alla quale possono accedere solamente le persone autorizzate da una determinata azienda che ne è gestore oltre che proprietaria. Con essa si ha l'esportazione all'esterno dei confini aziendali delle informazioni che sono reperibili sulla Intranet aziendale. La Extranet comprende le aziende e le persone che si trovano all'esterno del firewall con le quali l'azienda desidera comunque mantenere uno scambio di informazioni.

## 1.4 II WORLD WIDE WEB

Il *World Wide Web* (WWW o anche *WEB*) rappresenta uno spazio definito all'interno di Internet. Si usa la parola *spazio* perché è difficile descriverlo in altro modo. Infatti non si tratta di un oggetto fisico e nemmeno di un'entità riconducibile a confini geografici, bensì è un sistema di presentazione e soprattutto d'interconnessione tra le informazioni, concepito in modo da favorire il passaggio automatico da un documento all'altro e consentire la *navigazione* in un grande mare informativo, basandosi esclusivamente su ciò che appare sullo schermo del computer. Il World Wide Web fa parte di Internet ma comprende solo una parte delle risorse disponibili all'interno di quest'ultima, ecco perché lo definiamo come uno degli spazi contenuti all'interno del grande universo Internet.

Il progetto WEB è nato nel 1991 presso il CERN (*Consiglio Europeo per la Ricerca Nucleare*) di Ginevra quale sistema per semplificare lo scambio d'informazioni tra ricercatori scientifici che già utilizzavano Internet per far circolare i loro articoli e pubblicazioni. Il metodo che veniva usato era infatti basato sul protocollo FTP e presupponeva la conoscenza esatta della dislocazione fisica delle informazioni da scaricare dalla rete. Era inoltre essenziale possedere i programmi adatti ad "aprire" i documenti scaricati che potevano essere scritti con gli strumenti più diversi. Infine non era previsto nessun meccanismo di collegamento automatico tra le varie informazioni, cosa invece possibile con gli *ipertesti*, (come vedremo tra breve).

Nel 1989 Tim Berners-Lee, uno dei ricercatori del CERN, che può essere considerato il "padre" del WWW, stilò un documento in cui si chiedeva di creare un sistema che rendesse più rapida la condivisione d'informazioni, a livello mondiale, tra le diverse équipes di ricercatori che si occupavano di fisica nucleare per le alte energie, risolvendo i tre problemi prima evidenziati.

La proposta prevedeva tre componenti essenziali:

- un'interfaccia utente comune per tutti (indipendente quindi dal tipo di computer utilizzato),
- la capacità di incorporare nel sistema i tipi più diversi di documento e le tecnologie più varie,
- l'accessibilità universale a tutte le informazioni contenute in questo ambito.

Il nome stesso del progetto, WWW, sottolinea l'interconnessione tra informazioni, infatti la traduzione è *ragnatela estesa quanto il mondo*.

In sostanza si trattava di trasformare l'enorme capacità informativa disponibile su Internet in qualcosa che chiunque potesse usare e consultare con facilità.

Per questo fu necessario definire un nuovo linguaggio universale per la creazione di documenti con caratteristiche ipertestuali (il linguaggio *HTML* oggetto di queste dispense), una notazione universale di localizzazione degli stessi (gli *URL*, che illustreremo tra breve), un nuovo protocollo ottimizzato per il trasferimento di ipertesti (*l'HTTP*). L'architettura risultante è di tipo *client/server*. Il WEB infatti funziona grazie a dei computer, dotati di apposito software, chiamati *WEB server* su cui sono depositati documenti il cui formato e sistema di visualizzazione è conforme alle specifiche definite dal CERN; gli utenti possono poi usufruire di queste informazioni grazie a dei programmi client chiamati *browser* o *navigatori*.

Il progetto del CERN non fu subito considerato importante ma alla fine del 1990 iniziò il lavoro per sviluppare i primi strumenti World Wide Web. Essi si basavano ancora sull'interfaccia a caratteri tipica di Unix e del DOS, ma tra essi c'era anche il primo browser che si chiamava *www* e iniziò a essere utilizzato su scala ridotta nel marzo del 1991. Un'altra tappa determinante fu l'installazione di WAIS (*Wide Area Information Servers*) per consentire da World Wide Web la ricerca di documenti su scala mondiale usando



come parole chiave i termini contenuti nel testo dei documenti stessi. Verso la fine del 1991, il CERN annunciando l'esistenza del progetto a tutti gli scienziati impegnati nella ricerche sulla fisica nucleare per le alte energie, diede il "battesimo" ufficiale al World Wide Web.

Nel gennaio del 1993 erano già operativi 50 server WEB e alla fine dello stesso anno erano diventati 623.

La progressione negli anni seguenti è stata impressionante:

Periodo	Numero server WEB
gennaio 1993	50
dicembre 1993	623
giugno 1994	2.738
dicembre 1994	10.022
giugno 1995	23.500
gennaio 1996	90.000
giugno 1999	3.600.000

Riprendiamo la storia del World Wide Web con l'uscita del primo browser grafico per WWW: si chiamava Viola e funzionava su terminali X-Window cioè su terminali grafici collegati in rete locale che usano un particolare protocollo per accedere a un minicomputer dotato del sistema operativo Unix.

Nel 1993 l'NCSA (*National Center for Supercomputing Applications*) diffuse la prima versione di Mosaic per X-Window sviluppata da Marc Andreessen. Per molto tempo Mosaic è stato il principale browser per World Wide Web ed il suo impiego si è esteso alle piattaforme più diverse, tra cui anche Windows. Dal suo ceppo sono nati tutti gli altri browser oggi in circolazione.

Nel 1994 iniziarono a nascere le prime società di software specializzate nella produzione di prodotti per World Wide Web e per Internet. Una delle più importanti tra queste è la Mosaic Communication Corporation fondata in California dallo stesso Marc Andreessen che alla fine del 1994 rilasciò un browser per Windows, Macintosh e X Window che si chiamava Netscape. Il prodotto divenne talmente popolare da trasferire il proprio nome anche all'azienda medesima, che oggi si chiama appunto Netscape Communications.

Nel luglio del 1994 il CERN iniziò a trasferire il progetto WEB a un altro ente, il W3C (*World Wide Web Consortium*), al quale partecipa anche il MIT (*Massachusetts Institute of Technology*) e, per l'Europa, l'INRIA (*Institut National de Recherche en Informatique et en Automatique* - Istituto Nazionale della Ricerca sull'Informatica e sull'Automazione). Tale ente, libero e senza fini di lucro, è attualmente impegnato in attività finalizzate all'evoluzione delle tecnologie e nella definizione degli standard del WEB.

### 1.4.1 Ipertesti e Ipermedia

La proposta originariamente avanzata da Tim Berners-Lee descriveva in dettaglio l'utilità di avere un sistema che fosse facile da consultare su qualsiasi tipo di computer o terminale, nel quale fosse possibile eseguire ricerche e che creasse una connessione il più fitta possibile tra documenti pubblici e privati, al fine di facilitare e incoraggiare la navigazione tra questi ultimi. L'attuale forma del World Wide Web rispecchia queste specifiche iniziali e può essere suddivisa in tre componenti essenziali:

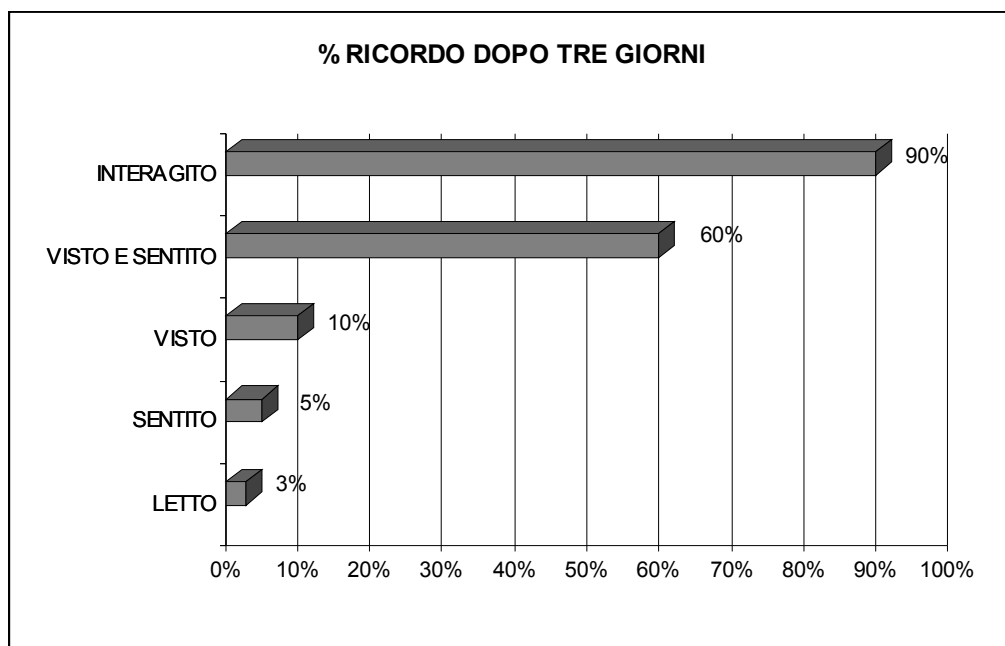
#### Internet - ipertesti - multimedialità

Con il termine MULTIMEDIALITA' si intende l'uso di una pluralità di mezzi di comunicazione (MEDIA) per la trasmissione di messaggi ed informazioni; i più importanti tra tali mezzi sono: testo scritto, suoni, immagini fisse o in movimento (filmati e animazioni grafiche).

E' noto che l'efficacia della trasmissione di un messaggio è legata alla ridondanza dei mezzi elementari con cui essa è effettuata. Pensiamo ad esempio alla narrazione, presente nella Bibbia, della consegna dei comandamenti a Mosè che avviene in mezzo a bufere, lampi, tuoni e forti sensazioni fisiche, oppure alla differenza che c'è tra leggere il testo di una commedia o vederla rappresentata, o ancora alla molteplicità di stimoli con cui vengono bombardati i ragazzi di oggi nelle discoteche.



Recenti studi dimostrano che, nella mente umana, il ricordo di una comunicazione persiste tanto più quanto maggiore è il numero dei MEDIA impiegati per trasmetterla e tale persistenza è massima se c'è interazione tra le parti coinvolte nella comunicazione.



Dal punto di vista informatico la multimedialità consiste in un insieme di tecnologie basate su un P.C. con le quali vengono combinati testo, immagini, suoni, filmati, per formare un unico messaggio ricco di informazioni ed efficace dal punto di vista della comunicazione. Più precisamente può essere chiamato **SISTEMA INTERATTIVO MULTIMEDIALE** un sistema di elaborazione in grado di utilizzare contemporaneamente **almeno tre** dei seguenti MEDIA: testo, grafici, animazioni grafiche, segnali audio, suoni musicali, voce, immagini, filmati.

La tecnologia oggi disponibile è comunque già in grado di assicurare l'integrazione di tutti questi media su una stessa macchina; si trovano infatti sul mercato personal computer equipaggiati con scheda audio, casse acustiche, scheda video in grado di gestire filmati, microfono, lettore di CD-ROM.

Fra le applicazioni multimediali meritano un'attenzione particolare gli **IPERMEDIA** che sono estensioni degli **IPERTESTI**.

Gli **IPERTESTI** nascono da un'idea di Ted Nelson che nel 1965 ipotizzò un sistema software in grado di memorizzare articoli, annotazioni, relazioni ed altre informazioni testuali nel quale l'utente poteva navigare liberamente.

Alla base dell'idea di Nelson c'era l'osservazione che l'uomo parla in modo sequenziale perché è dotato di un solo tratto vocale, legge e scrive sequenzialmente perché in tal modo sono organizzati i testi, ma ha una capacità di pensiero molto più sofisticata. Infatti la nostra mente opera spesso per associazione di idee ed è in grado di costruire una vera e propria "rete" di conoscenze molto complessa in cui molti elementi sono in relazione non lineare tra loro.

Gran parte dei sistemi informatici forniscono strumenti, anche sofisticati, in grado di gestire e manipolare dati ed informazioni in modo solo sequenziale e senza alcuna visione integrata. Gli ipertesti invece permettono di stabilire collegamenti e rimandi all'interno dei documenti o fra documenti diversi creando una organizzazione non lineare del testo che non può essere "resa" su una singola pagina; si aggiunge in pratica una "terza dimensione" al testo in modo da "entrare dentro" di esso.

Pensiamo ad esempio di avere un testo riguardante Foscolo e leggiamo che in un certo periodo della sua vita egli fu ammiratore di Napoleone; se vogliamo maggiori informazioni su quest'ultimo dobbiamo cambiare testo e passare ad uno che lo riguardi. Con un ipertesto invece si può passare con un semplice comando (un "click" del mouse sul nome Napoleone) alle informazioni che ci interessano. Inoltre possiamo continuare la nostra "navigazione" nel documento in modo molto libero sfruttando i collegamenti in esso presenti ed





arrivando ad esempio alla Napoli dei primi anni dell'800 passando per Gioacchino Murat, cognato di Napoleone, che fu re nella città in quel periodo.

Tecnicamente gli ipertesti si basano su grafi (reti) di nodi (concetti, idee, documenti) tra loro collegati e vengono creati con appositi programmi e con tecniche di cui comunque l'utilizzatore non deve preoccuparsi. I collegamenti tra i vari documenti avvengono grazie a dei *riferimenti*. Il riferimento (detto *iperlink* o *collegamento ipertestuale*) deve essere evidenziato in qualche modo rispetto al resto del testo, ad esempio con un colore diverso o con la sottolineatura.

Si parla di IPERMEDIA quando in un ipertesto vengono integrati suoni, immagini, filmati ed altro ancora. Nell'esempio precedente si potrebbe pensare di avere a disposizione anche l'immagine digitalizzata di un quadro raffigurante Napoleone oppure un paesaggio della Napoli di inizio '800 o, ancora, l'attacco della Terza Sinfonia di Beethoven che era stata appunto dedicata al Bonaparte.

## 1.4.2 Come funziona il WWW

Come evidenziato in precedenza gli ipertesti e la multimedialità (e quindi gli ipermedia) sono componenti essenziali del WWW che, non a caso, viene anche definito *iperspazio*.

In essi l'elemento fondamentale è l'iperlink attraverso il quale si può passare ad un altro documento che può trovarsi sul nostro computer, sul server a cui siamo già collegati o addirittura su un altro server situato in qualunque parte del mondo (purché collegato ad Internet). Il salto può comunque anche avvenire verso un'altra sezione dello stesso documento in cui già ci troviamo.

Se il sistema ipertestuale è ben fatto il documento in cui si arriva dovrebbe contenere un collegamento a quello di provenienza così che sia sempre possibile ritornare sui propri passi. Poiché nel mondo WWW questo viene fatto di rado, è compito del browser compensare questa mancanza tenendo traccia del percorso e permettendo in qualsiasi momento di muoversi a ritroso (tasto *BACK*). Parlando di documenti multimediali il collegamento può portarci in ogni genere di contenitore e produrre la risposta più diversa. Per il momento, una buona parte dei collegamenti che si trovano sul World Wide Web rimandano a pagine di testo oppure a elementi grafici statici; sempre più spesso però si hanno connessioni con veri e propri programmi scritti con linguaggi come il PERL o JAVA che permettono di realizzare e visualizzare pagine WEB "dinamiche".

L'elemento cardine per il funzionamento del World Wide Web è l'HTTP (*Hypertext Transfer Protocol*) che è un protocollo client-server, basato su TCP/IP per lo scambio dei documenti ipertestuali. Si tratta di un protocollo molto semplice che regola l'interazione tra il nostro browser e il particolare server con cui quest'ultimo di volta in volta si connette a seguito di una nostra richiesta diretta oppure seguendo un collegamento ipertestuale.

La transazione tra queste due entità si svolge in quattro fasi:

connessione - richiesta del documento - risposta – disconnessione

Le prime tre sono di solito segnalate nella finestra del browser con scritte che ci spiegano cosa sta succedendo. Nella richiesta il browser deve specificare al server quale protocollo deve essere utilizzato (HTTP oppure FTP o altro ancora) in quanto lo stesso browser può essere utilizzato per collegarsi anche con server che non fanno parte del World Wide Web e che offrono servizi Internet più tradizionali, come appunto lo scaricamento di file attraverso il protocollo FTP.

Dopo che la pagina è giunta sulla nostra macchina, la connessione col server s'interrompe e va ripresa quando si chiede di passare a una seconda pagina sul medesimo server oppure a un secondo server. Questo talvolta vale anche per il percorso a ritroso, dove si chiede di visualizzare una seconda volta una pagina che avevamo già visto.

L'eventuale presenza di grafica rende questo andirivieni abbastanza lento poiché tutte le volte gli elementi grafici (spesso "grandi" diversi Kbyte o Mbyte) devono essere scaricati di nuovo. Una soluzione pratica per eliminare questo inconveniente consiste nel creare una *cache* (memoria di transito) sul disco della nostra macchina o sul server che fa da *gateway* (porta di accesso verso Internet). Così, ogni volta che chiediamo una pagina già vista, la caricheremo dal nostro disco locale oppure dal disco del server vicino anziché richiamarla dalla rete.



Un sistema per non dover scaricare continuamente la stessa pagina consiste nel salvarla sulla propria macchina come file a sé stante e nel richiamarla dall'interno del browser come si richiamerebbe un qualsiasi documento dall'interno di un word processor (tra l'altro, alcuni programmi di elaborazione dei testi permettono di visualizzare e modificare queste pagine scaricate in locale senza dover ricorrere al browser).

## 1.5 Gli indirizzi delle risorse di Internet

### 1.5.1 Indirizzi IP e URL (o URI)

Abbiamo visto come, con il protocollo HTTP, si possa accedere alle informazioni presenti in Internet; questo però è possibile solo se si conosce l'indirizzo della risorsa (pagina WEB o altro) a cui si vuole fare riferimento e cioè il suo URL (*Uniform Resource Locator*) o URI (*Uniform Resource Identifier*).

Ogni URL della rete identifica una certa risorsa e non possono esistere due risorse diverse con stesso indirizzo.

Spesso si dice anche che un URL identifica un *sito Internet* dove con *sito* si intende la sezione del disco di un particolare computer nella quale risiedono i documenti WEB; il significato viene però solitamente esteso fino a comprendere anche la macchina nel suo complesso e l'organizzazione che la usa per pubblicare le proprie informazioni in Internet.

Un URL è una stringa che inizia con il nome del protocollo da utilizzare per reperire la risorsa (esempio `http://`). Per illustrare completamente la sua struttura occorre però fare prima riferimento al modo in cui i computer sono identificati in Internet (o in qualsiasi rete basata su protocollo TCP/IP): ogni computer è individuato univocamente da un *indirizzo IP* o *IP-address* composto da quattro ottetti di bit (cioè da quattro gruppi di 8 bit).

Ad esempio il server WEB della casa editrice Mondadori è su un computer che ha indirizzo:

11000010.10001100.11100000.10100001

Come si vede l'IP-address è espresso con quattro numeri binari; ciò si giustifica in quanto sappiamo benissimo che il linguaggio dei computer è il sistema binario costituito dai soli simboli 0 e 1. Per comodità però gli indirizzi IP vengono sempre indicati "tradotti" in decimale e così il precedente diventa:

194.140.224.161

Gli indirizzi IP sono suddivisi in classi e ne esistono alcuni che sono riservati per usi speciali (ad esempio quelli che iniziano con 192.168 e che sono usati per le reti private locali); questi argomenti non vengono però approfonditi in questa sede.

L'assegnazione degli indirizzi di rete viene curata da una organizzazione senza fini di lucro, l'ICANN (*Internet Corporation for Assigned Names and Numbers*, [www.icann.org](http://www.icann.org)), la quale a sua volta delega ad enti nazionali la gestione degli indirizzi di rete nei vari paesi. In Italia tale gestione è stata curata fino al 1998 dal GARR (*Gruppo Armonizzazione delle Reti di Ricerca*) e adesso dalla *Registration Authority* italiana, che fa capo al CNR (*Consiglio Nazionale delle Ricerche*), in base alle indicazioni fornite dalla *Naming Authority* italiana (che opera in stretto rapporto con il Ministero delle poste e delle telecomunicazioni). Maggiori dettagli sull'assegnazione degli indirizzi IP si possono trovare all'indirizzo <http://www.nic.it>.

Un difetto del complesso, ma efficiente, metodo di indirizzamento degli Host di Internet è che gli indirizzi sono limitati e con gli attuali ritmi di crescita della rete si corre seriamente il rischio di esaurire entro poco tempo tutti gli indirizzi disponibili. Per questa ragione è stata sviluppata recentemente una versione evoluta del protocollo IP, denominata "IP Next Generation" o "IPV6", basata su un sistema di indirizzamento a 128 bit che assicurerà un massiccio incremento nella disponibilità di indirizzi di rete.

L'indirizzo numerico comunque non è utilizzabile comodamente e non è facile da ricordare; sarebbe molto meglio poter individuare i vari computer con un nome. Questo è possibile grazie all'introduzione nella rete del DNS (*Domain Name Server*).



Il DNS è il meccanismo con cui si riesce a indirizzare le risorse su Internet utilizzando una notazione mnemonica, allettante anche dal punto di vista commerciale, garantendo al tempo stesso una individuazione univoca della risorsa sulla rete.

Attraverso il DNS ogni host di Internet può essere dotato di un nome composto da stringhe di caratteri. Tali stringhe, a differenza dell'indirizzo numerico, possono essere di lunghezza illimitata.

L'indirizzo del server della Mondadori diviene dunque *www.mondadori.com* che è senz'altro più comodo da utilizzare e da ricordare (oltre che più gradito alla stessa azienda Mondadori).

Come si può vedere, anche i nomi mnemonici sono sequenze di simboli separati da punti e questo rispecchia la struttura gerarchica del DNS. Esso infatti suddivide l'intera rete in settori, denominati *domini*, a loro volta divisi in *sottodomini*, e così via per vari livelli; ogni sottodominio fa parte del dominio gerarchicamente superiore: all'ultimo livello della gerarchia ci sono i singoli computer.

L'identificativo di un host riassume le varie gerarchie di domini a cui appartiene: illustriamo il concetto servendoci di un altro esempio di nome mnemonico:

*www.prog-aut.itis.biella.it*

in realtà il nome del computer è solo *www*, il resto della stringa serve ad indicare chi ha la responsabilità di tale computer e del suo nome. Scorrendo la stringa da destra a sinistra troviamo *it* che è il *dominio di primo livello* e sta ad indicare che il computer si trova in una gerarchia facente capo ad una autorità nazionale italiana (*it* sta per Italy). Successivamente abbiamo *biella* che è un *sottodominio di primo livello* ed indica che l'autorità facente capo alla città di Biella ha ricevuto il permesso da quella immediatamente superiore (in questo da quella del dominio *it*) di poter a sua volta concedere sottodomini ad altre autorità sottostanti o nomi ai propri computer. Continuando troviamo *itis* che è un *sottodominio di secondo livello* che viene gestito da una autorità per questo delegata da quella di livello superiore (in questo caso *biella*). Ancora più a sinistra troviamo *prog-aut* che è un *sottodominio di terzo livello* gestito da una autorità gerarchicamente sottostante a quella che gestisce il sottodominio *itis*. Questa autorità ha deciso di chiamare *www* il computer che contiene la pagina iniziale (*home page*) del sito in questione.

Naturalmente non tutti gli identificativi sono così articolati; ad esempio in:

*www.linux.it*

abbiamo solo il sottodominio di primo livello *linux*, e l'autorità che lo gestisce ha deciso di chiamare *www* il computer che ospita la home page del sito.

Il numero e le sigle dei domini di primo livello, o *domini principali*, sono fissati a livello internazionale e vengono gestiti da appositi organismi. Nell'ambito di ognuno di tali domini possono essere creati un numero qualsiasi di sottodomini rispettando però le regole stabilite da ogni autorità nazionale di gestione del DNS.

Quando il DNS è stato sviluppato, Internet era diffusa, salvo rare eccezioni, solo negli Stati Uniti e la rete venne suddivisa in sei domini principali, tuttora esistenti, le cui sigle caratterizzano il tipo di ente o organizzazione che possiede un certo sito:

- EDU per gli enti di ricerca e università
- COM per le organizzazioni commerciali
- GOV per gli enti governativi
- MIL per gli enti militari
- NET per gli enti di gestione della rete
- ORG per gli enti diversi (volontariato, associazioni senza fini di lucro)

Quando la rete ha cominciato a diffondersi a livello internazionale sono stati creati altri domini principali, uno per ogni nazione, ad esempio:

- IT per l'Italia
- UK per l'Inghilterra
- FR per la Francia
- DE per la Germania



Inoltre talvolta anche negli Stati Uniti si usano suffissi geografici.

Vediamo altri esempi di URL completi:

<http://www.tin.it/> è l'indirizzo del sito della Telecom Italia Net

<http://www.deejay.it/> è l'indirizzo del sito di Radio DeeJay

<http://www.cambridge.edu/> è l'indirizzo del sito dell'università di Cambridge negli USA

<http://www.fbi.gov/> è l'indirizzo del sito dell'FBI

<http://www.ci.berkeley.ca.us/> è l'indirizzo della rete civica della città di Berkeley in California.

Dal punto di vista tecnico il DNS è costituito da un sistema di archivi distribuiti nella rete e collegati tra loro chiamati "name server". Essi svolgono la funzione di tradurre i nomi in indirizzi numerici (tecnicamente si parla di "risoluzione dei nomi") per conto degli host o di altri name server. Infatti la comunicazione effettiva tra i computer della rete avviene sempre e solo attraverso gli indirizzi IP numerici.

Quando un computer deve collegarsi ad un altro, con nome ad esempio [www.linux.it](http://www.linux.it), esso interroga il proprio name server locale per risolvere il nome in questione (è per questo che quando ci abboniamo con un fornitore di servizio, o "provider" per navigare in Internet, fra i parametri che ci vengono forniti per impostare il browser c'è anche l'indirizzo IP di un server DNS).

Nel caso il name server non sia in grado di risolvere il nome richiesto, chiede "aiuto" ad un altro server, detto *name server di primo livello* la cui scelta è determinata dal dominio principale dell'indirizzo in questione. Questo server, a sua volta, può non essere in grado di rispondere e ricorre quindi ad altri name server (di livello inferiore). Il procedimento continua fino al reperimento dell'indirizzo del computer cercato, se esiste.

Nello svolgere questo compito il name server memorizza gli indirizzi che ha conosciuto in modo da rispondere più velocemente a successive richieste.

Grazie a questo meccanismo il DNS è sempre aggiornato in modo automatico e non è necessaria alcuna autorità centrale che memorizzi nomi ed indirizzi dei milioni di computer collegati ad Internet.

Come avviene per gli indirizzi IP, la gestione del DNS in un dominio di primo livello viene affidata a degli enti specifici. Questi enti hanno il compito di assegnare i nomi di sottodominio, controllando che non ci siano omonimie, e di gestire l'archivio principale del dominio di cui sono responsabili. In Italia l'ente che si occupa di questo è ancora la Registration Authority all'indirizzo [www.register.it](http://www.register.it); negli Stati Uniti la gestione dei nomi è affidata a delle compagnie private sotto il controllo della già citata ICANN.

Per concludere notiamo che, usando i browser più recenti, possiamo scrivere gli URL tralasciando il protocollo (ammesso che sia http) in quanto viene assegnato automaticamente dal programma come <http://>; inoltre nessuno vieta l'uso di indirizzi numerici per fare riferimento ad un certo sito anche se è molto più difficile ricordarli; ad esempio per collegarsi al sito dell'FBI si potrebbe anche digitare l'URL: <http://199.170.0.150/>

## 1.5.2 URL con percorsi e nomi di file

Gli esempi visti sinora si riferiscono sempre all'indirizzo della home page dei siti considerati; se invece vogliamo visualizzare con il nostro browser una pagina particolare, contenuta in una certa directory del server, dovremo aggiungere all'URL il percorso completo delle directory ed il nome del file che contiene la pagina desiderata.

Ad esempio: [http://www.meteo.fr/tpsreel/e\\_tpsre.html](http://www.meteo.fr/tpsreel/e_tpsre.html) fa riferimento alla pagina contenuta nel file di nome *e\_tpsre.html* nella directory *tpsreel* del server *www* del sito meteo della Francia (è una pagina che contiene le immagini dell'europa inviate dal satellite per le previsioni meteo).

Si noti come nell'indicazione del percorso si usa il simbolo "/" per indicare le sottodirectory, anziché il simbolo "\" come si fa, ad esempio, in MS-DOS.

L'importante comunque è conoscere l'URL della pagina iniziale di un sito in quanto da essa, seguendo i vari collegamenti (*iperlink*), sarà possibile rintracciare le altre pagine che fanno parte del sito.

## 1.5.3 URL con altri protocolli

Gli URL permettono di individuare risorse Internet generiche e non solo siti WEB come negli esempi visti sinora; è infatti possibile inserire altri protocolli al posto di <http://> come ad esempio <ftp://>. Questo è il



protocollo usato per il trasferimento di file da un computer ad un altro. Di solito i siti FTP si trovano su macchine che si chiamano ftp (e non www) e allora i loro URL saranno simili al seguente:

*ftp://ftp.winsite.com/*

Altri protocolli come *gopher://*, *mailto:*, *news:* sono molto meno usati. Importante è invece la stringa *file://* (usata al posto del protocollo anche se non è un protocollo); con essa si vuole fare riferimento a dei file contenuti sul computer *locale*, cioè quello che stiamo usando con il browser per "navigare" in Internet. Ad esempio l'URL *file://esempi/indice.htm* ci collega con la pagina contenuta nel file di nome *indice.htm*, che è nella directory *esempi* del nostro computer.

## 1.6 Collegarsi a Internet da casa

Per collegarsi a Internet sono necessari (oltre naturalmente ad un computer) un modem ed un abbonamento ad un *ISP* (*Internet Service Provider*). Un ISP è una azienda che fornisce a pagamento o, ultimamente anche gratis, la possibilità di collegarsi alla rete, di avere una o più caselle di posta elettronica e, talvolta, anche un pò di spazio sul proprio server per pubblicare pagine in Internet. In Italia possiamo citare tra i provider più noti *TIN* (*Telecom Italia Net*), *Libero*, *Tiscali*.

Si deve inoltre avere il protocollo TCP/IP sul proprio computer ed anche il protocollo *PPP* (*Point to Point Protocol*). Questo non è un problema visto che questi protocolli sono forniti a corredo di tutti i sistemi operativi più diffusi per Personal Computer.

Il protocollo PPP consente di usare i protocolli Internet (IP), normalmente utilizzati su connessioni *Ethernet*, cioè in reti locali, su linee seriali e quindi per i collegamenti via modem attraverso la porta seriale del personal computer (*RS-232*).

Il **MODEM** (**MOD**ulatore **DEM**odulatore), è una periferica che permette il collegamento tra computer fisicamente distanti tra loro usando le normali linee telefoniche in quanto trasforma (in un modo che qui non approfondiamo) i segnali digitali (bit), propri degli elaboratori, in segnali analogici adatti a essere trasportati su tali linee (originariamente progettate per la comunicazione vocale).

Se due computer distanti (chiamiamoli C1 e C2) si devono scambiare informazioni saranno necessari due modem (rispettivamente M1 e M2) uno per ogni elaboratore collegati alla linea telefonica; se il computer C1 invia un messaggio questo sarà convertito (modulato) in analogico da M1 e instradato sulla linea; all'arrivo presso C2 il messaggio viene riconvertito (demodolato) in digitale da M2. Se è C2 a inviare un messaggio i ruoli dei dispositivi sono naturalmente invertiti.

I modem si classificano secondo la loro velocità, misurata in base ai bit al secondo (bps) che riescono ad inviare o ricevere; gli apparecchi di ultima generazione riescono ad arrivare a 55.600 bps ed i loro prezzi sono abbordabili (molto sotto al mezzo milione).

Negli ultimi tempi è possibile collegarsi al provider anche con la linea ISDN, usufruendo di una velocità di 64.000 o 128.000 bps, grazie ad un dispositivo chiamato *modem ISDN*. In questo caso però il nome non è corretto in quanto si tratta solo di un adattatore e non di un vero e proprio modem visto che la linea ISDN, pur sfruttando i normali cavi telefonici, è digitale e non analogica. Anche i prezzi degli adattatori ISDN sono ormai molto bassi e paragonabili a quelli dei modem.

A proposito di costi è bene ricordare come sia fondamentale abbonarsi ad un ISP che abbia almeno un *POP* (*Point Of Presence*), cioè un server a cui collegarsi via modem e telefono, nella propria città in modo da usufruire della tariffa telefonica urbana; diversamente la "navigazione nell'iperspazio" potrebbe causare forti aumenti della bolletta telefonica.



## 2 IL LINGUAGGIO HTML

### 2.1 Introduzione

I documenti presenti nel WEB hanno un formato particolare e usano al proprio interno una serie di codici che dicono al browser come visualizzare il testo e le immagini che vi sono associate. Il più diffuso tra i linguaggi usati per questa codifica è l'HTML (*Hypertext Markup Language - linguaggio per la codifica degli ipertesti attraverso marcatori*) che costituisce una versione semplificata dell'SGML (*Standard Generalized Markup Language - linguaggio di codifica standard e generalizzato*).

L'SGML è stato sviluppato dall'organizzazione internazionale per gli standard nel 1986 per definire linguaggi markup progettati per vari scopi diversi. Ogni linguaggio della famiglia SGML deve rispettare certi requisiti fra i quali quello che tutti i simboli siano definiti e descritti usando un *DTD* (*Document Type Definition*); il DTD per l'HTML definisce i marcatori disponibili e il modo di usarli.

L'HTML non è un linguaggio di programmazione e un documento scritto in HTML non è assolutamente un programma cioè una serie di istruzioni da eseguire su dei dati. Piuttosto il documento HTML è esso stesso un dato ed il linguaggio definisce le regole per l'inserimento di particolari *TAG* (*marcatori*) che indicano ai browser quale struttura avrà la pagina da visualizzare.

La funzione principale dell'HTML è quella di classificare le varie parti che compongono un documento: si può indicare quale parte rappresenta il titolo, in quali posizioni inserire delle immagini, quali parti enfaticizzare e così via. Sono presenti anche alcuni comandi di formattazione e layout ma queste funzioni riguardanti l'aspetto esteriore del documento sono secondarie rispetto alla descrizione della struttura generale dei suoi contenuti.

Un aspetto importantissimo dell'HTML è che è indipendente da qualsiasi piattaforma Hardware e software: in altre parole una volta scritto un documento con gli elementi standard di HTML, si può essere sicuri che la pagina verrà visualizzata nello stesso modo con qualsiasi browser su qualsiasi computer. Inoltre il suo utilizzo è libero, non ci sono licenze né aggiornamenti da comprare e non si dipende da nessuna azienda produttrice di software.

### 2.2 Storia dell'HTML

L'HTML si sta evolvendo rapidamente e ne sono state già rilasciate alcune versioni. E' stato ideato nel 1989 insieme al WWW da Tim Berners Lee. Attualmente la definizione degli standard dell'HTML (oltre che di quelli dell'HTTP e di altre tecnologie WEB) è sotto la responsabilità del consorzio W3C.

La prima versione dell'HTML si chiamava HTML (senza numero di versione) e non ha avuto una grande diffusione perché quando è apparsa esistevano pochissimi server WEB. E' comunque servita da base per le successive versioni che hanno sempre conservato la compatibilità all'indietro (i documenti scritti con la prima versione possono essere tranquillamente usati con le versioni più recenti).

Nel 1993 Dave Ragget ha sviluppato una versione aggiornata di HTML, chiamata HTML+ che non è mai diventato uno standard ufficiale ma le cui innovazioni sono state incorporate nella versione 2.0.

Alla fine del 1994 è stato approvato lo standard HTML 2.0, più affidabile e un po' più semplice delle versioni precedenti. Esso ha conosciuto una grande diffusione anche se è stato criticato dagli sviluppatori di siti WEB perché permetteva l'utilizzo di un numero troppo esiguo di comandi di formattazione del documento.

Per questo motivo, e anche per la lentezza con la quale il W3C approvava i nuovi standard, le società produttrici di programmi browser (principalmente Netscape e Microsoft) hanno iniziato a supportare tag non standard e non approvati dal W3C, conosciuti come *estensioni* per l'HTML. Naturalmente ogni browser gestiva le proprie estensioni e i programmi rimasti "fedeli" agli standard non erano in grado di interpretare i nuovi tag.

Nel 1995 è stato redatto lo standard HTML 3.0 che però prevedeva troppi cambiamenti rispetto alla versione 2.0 e quindi non è stato preso in considerazione dagli sviluppatori.

Migliore fortuna ha avuto invece la versione HTML 3.2 del 1996 che era maggiormente compatibile con la versione 2.0 e comprendeva anche le estensioni usate dai browser più diffusi (NETSCAPE NAVIGATOR e INTERNET EXPLORER di Microsoft).



La versione più recente di HTML è la 4.0 che rispetto alla 3.2 incorpora alcune funzioni che erano estensioni per la 3.2, gestisce un insieme di caratteri (chiamato UNICODE) più esteso del precedente (chiamato LATIN-1) ed anche un nuovo marcatore per gli oggetti multimediali.

Occorre comunque osservare che la versione di HTML supportata sicuramente anche dai browser di minore diffusione è la 3.2.

## 2.3 Aspetto di un file HTML

Le pagine scritte in HTML sono costituite da puro testo in formato *ASCII*, non contengono informazioni specifiche di una certa piattaforma o di un certo programma e possono essere lette o modificate con qualsiasi editor di testo (ad esempio *EDIT* del MS-DOS o *vi* di LINUX). Al loro interno si trovano due tipi di oggetti:

il testo del documento  
i TAG HTML

Un tag (o *elemento HTML*) è un insieme di simboli con un significato speciale: inizia con il simbolo di minore (<), continua con una *parola riservata* e termina con il segno di maggiore (>). I seguenti sono esempi di tag:

<html>                      </B>                      <BODY>                      <P>

Nei tag non si ha distinzione tra lettere maiuscole e minuscole, quindi <BODY>, <body>, <Body> rappresentano lo stesso tag.

Esistono due tipi di tag: quelli di inizio con i quali si attivano certe opzioni o funzioni, e quelli di fine con i quali si disattivano le stesse opzioni e funzioni. Compreso tra essi c'è il *contenuto* dell'elemento HTML contrassegnato dai tag in questione.

I marcatori di fine si scrivono nello stesso modo di quelli di inizio eccetto per il fatto che dopo il simbolo di minore (<) si ha sempre il simbolo della barra (/). Se ad esempio scriviamo nel documento HTML:

<B>Questo testo sarà in grassetto</B>

il testo compreso tra i due tag, (cioè il contenuto di quell'elemento) sarà visualizzato in grassetto (B sta per Bold cioè, appunto, grassetto).

Esistono anche degli elementi che non prevedono obbligatoriamente il tag di fine come <P>, che indica l'inizio dell'elemento paragrafo ed il cui contenuto è il testo del paragrafo stesso. Inoltre ci sono degli elementi senza neanche il contenuto come <HR>, che provoca l'inserimento nella pagina di una riga orizzontale.

Alcuni tag prevedono anche degli *attributi*, cioè degli elementi opzionali che definiscono le modalità di funzionamento di quell'elemento HTML.

La cosa più importante da capire riguardo ai tag è che essi non vengono mai visualizzati dal browser; viene invece visualizzato il loro "effetto". Inoltre se qualche tag è scritto male, contiene errori di sintassi o non è riconosciuto per qualsiasi altro motivo, viene semplicemente ignorato dal browser che non dà nessuna segnalazione di errore al riguardo.

## 2.4 Strumenti per la creazione di documenti HTML

Per scrivere un documento in linguaggio HTML, come detto, si può usare un comunissimo e semplicissimo editor di testo. Esistono però anche altri strumenti più sofisticati che si suddividono fondamentalmente in due categorie:

- editor studiati appositamente per la scrittura di codice HTML ma a formattazione differita (il risultato della formattazione del documento è visibile solo successivamente usando un browser).
- editor a formattazione immediata (il risultato finale della formattazione del documento è visibile immediatamente) che creano automaticamente il codice HTML;



Nel primo caso si tratta di editor con delle funzioni aggiuntive per la scrittura degli elementi HTML, per il controllo ortografico, per l'ordinamento del testo ed altre ancora.

Nel secondo caso si hanno strumenti come NETSCAPE COMPOSER o MICROSOFT FRONTPAGE che permettono di comporre la pagina partendo dal suo aspetto e generano automaticamente il sorgente non richiedendo quindi alcuna conoscenza del linguaggio HTML. Questi programmi sono detti anche editor WYSIWYG (*What You See Is What You Get, ciò che vedi è ciò che ottieni*) ed il loro uso può apparire conveniente tanto da far ritenere superfluo un corso di HTML. I fautori di tali strumenti affermano tra l'altro che con essi i creatori di pagine WEB sono sollevati dalla necessità di imparare comandi molto complicati e che la produttività e la qualità del lavoro sono migliori. Si devono però anche considerare dei punti a sfavore non trascurabili: intanto le regole dell'HTML sono molto semplici e lineari; inoltre se si conosce il linguaggio si hanno maggiori possibilità di correggere errori e risolvere problemi sfruttando a pieno le sue potenzialità e si può intervenire sui propri documenti da qualsiasi computer e con qualunque editor; infine il codice HTML creato automaticamente è solitamente di cattiva qualità, non efficiente, male organizzato e può contenere elementi incompatibili con gli standard ufficiali del linguaggio.

Quindi è utile conoscere l'HTML e per la creazione di documenti non è consigliabile l'uso di programmi troppo sofisticati; bastano i semplici editor come EDIT del MS-DOS, vi di LINUX, NOTEPAD di WINDOWS o al massimo gli editor specifici per l'HTML.

## 2.5 Struttura di un documento HTML

La struttura generale di un documento HTML viene definita con i seguenti elementi fondamentali:

```
html
head
body
title
```

Il solo elemento <title> è obbligatorio in quanto gli altri, se assenti, vengono automaticamente inseriti da molti browser; è comunque buona norma includere tutti e quattro questi elementi in qualsiasi documento HTML.

All'inizio del sorgente HTML si deve inoltre inserire una riga che specifichi quale è la versione di HTML utilizzata. Per fare questo si usa la dichiarazione <!DOCTYPE> che non è un tag HTML ma una entità SGML.

Molti browser gestiscono tranquillamente anche documenti WEB mancanti di tale elemento ma è comunque opportuno inserirlo per evitare incompatibilità.

Per la versione HTML 3.2 la linea da scrivere è la seguente (vengono omissi i dettagli sul significato dei singoli elementi di tale dichiarazione):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

Essendo essa abbastanza complessa e da ricordare può essere conveniente crearsi un modello di documento HTML, che includa solo tale linea, da riempire di volta in volta con il contenuto desiderato.

### 2.5.1 L'elemento HTML



Lo scopo di questo elemento è quello di indicare che il file che si sta scrivendo è realizzato in linguaggio HTML. Tutto il testo, comprensivo degli altri tag, dovrà essere racchiuso tra i tag di inizio e fine dell'elemento html:

```
<HTML>
....
.... documento
....
</HTML>
```

Un documento HTML deve sempre contenere due sezioni o parti:





-  la sezione HEAD che contiene elementi che non influenzano la visualizzazione da parte dei browser;
-  la sezione BODY che contiene il documento vero e proprio.

### 2.5.2 L'elemento HEAD

La sezione HEAD inizia con il tag `<HEAD>` e termina con `</HEAD>` e contiene di solito solo il titolo del documento racchiuso tra i tag `<TITLE>` e `</TITLE>` anche se può contenere altre informazioni come i dati sull'autore e sul programma che ha generato il documento, o anche degli *script* cioè insiemi di comandi scritti in un qualche linguaggio apposito e che possono svolgere varie attività interagendo con il browser.

```
<HEAD>
<TITLE>Istituto F. Besta</TITLE>
</HEAD>
```

Il titolo deve essere una descrizione del documento breve, perché viene visualizzato dai browser nella barra del titolo della finestra, e significativa, perché viene utilizzato dai programmi che catalogano i documenti di Internet per creare indici di ricerca.

### 2.5.3 L'elemento BODY

La sezione BODY è racchiusa tra i tag `<BODY>` e `</BODY>` e contiene la parte del documento che viene visualizzata dai browser quando si apre la pagina.

Il seguente è un semplicissimo esempio che riassume la struttura generale di una pagina HTML con l'uso dei tag sinora illustrati:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">

<HTML>
<HEAD>
<TITLE>Istituto F. Besta – Esempio in HTML</TITLE>
</HEAD>
<BODY>
Questo è il testo che viene visualizzato
</BODY>
</HTML>
```

### 2.5.4 Elementi di blocco e elementi di testo

All'interno della sezione body possono essere presenti molti elementi suddivisi fondamentalmente secondo due tipologie:

- elementi di testo
- elementi di blocco

I primi si usano per inserire immagini, creare collegamenti, modificare l'aspetto del testo, i secondi per definire gruppi di testo con uno scopo specifico, come intestazioni, tabelle ed altro ancora.

La differenza principale tra i due tipi di elementi è che quelli di blocco provocano interruzioni di paragrafo e quelli di testo invece no.

### 2.5.5 Intestazioni

Le intestazioni sono elementi DI BLOCCO e ne esistono sei livelli diversi. Non è obbligatorio usarli ma sono utili perché aiutano a organizzare in modo razionale le pagine.

Ogni intestazione inizia con `<Hn>` e termina con `</Hn>` dove n è il numero corrispondente al livello desiderato. Nei tag di inizio è possibile inserire i seguenti attributi:



ALIGN="LEFT"  
ALIGN="RIGHT"  
ALIGN="CENTER"  
ALIGN="JUSTIFY" (novità dell'HTML 4.0) e quindi non supportata da tutti i browser

Il loro effetto è rispettivamente quello di allineare le intestazioni a sinistra, destra, centro e ad entrambi i margini.

## 2.5.6 Paragrafi

In HTML l'inizio di un paragrafo è indicato con il tag <P> mentre non è obbligatorio il tag di fine </P>. E' molto importante notare che la presenza di spazi o righe vuote nel sorgente HTML viene ignorata dai browser (gli spazi vengono compattati) e quindi l'unico modo per separare correttamente i paragrafi è, appunto, l'uso del tag <P>.

Per chiarire meglio il discorso consideriamo le seguenti porzioni di codice HTML:

```
<H1>Semplice esempio</H1>  
<P> Questo è un semplice esempio con due righe di testo.  
<P> Questa è la seconda riga
```

oppure

```
<H1>  
Semplice esempio  
</H1>  
<P> Questo è un semplice esempio con due righe di testo.  
  
<P> Questa è la seconda riga
```

oppure

```
<H1>Semplice                esempio  
</H1>  
<P> Questo è un semplice  
esempio con due righe di testo.  
<P> Questa è la seconda riga
```

questi tre esempi produrranno esattamente lo stesso effetto al momento della visualizzazione con il browser.

A proposito dei paragrafi si deve anche ricordare che non è opportuno cercare di inserire righe vuote con una sequenza di <P> consecutivi perché molti browser in questo caso li riducono ad un solo <P> annullando lo sforzo compiuto dall'autore del sorgente HTML.

All'interno dell'elemento paragrafo si possono utilizzare gli attributi di allineamento visti per le intestazioni. Naturalmente il tag <P> è un elemento DI BLOCCO.

## 2.5.7 Righello orizzontale

Con il tag <HR> si inserisce nel documento una riga orizzontale, oltre a separare il testo in due paragrafi separati. Questo elemento è DI BLOCCO ed è sempre gestito anche se non tutti i browser visualizzano la riga nello stesso modo.

## 2.5.8 Centatura

I tag <CENTER> e </CENTER> vengono usati per centrare tutto ciò che è contenuto tra essi, testo o immagini. Anche questo è un elemento DI BLOCCO.



## 2.5.9 Interruzione di linea

Per interrompere una linea e "tornare a capo" si può usare il tag `<BR>`. Questo elemento è DI TESTO e si differenzia da `<P>` perché quest'ultimo, oltre a terminare la linea termina anche il paragrafo iniziandone uno nuovo.

## 2.5.10 Grassetto, corsivo, sottolineato

Con i tag `<B>` e `</B>` si enfatizza il testo compreso, con `<I>` e `</I>` si visualizza in corsivo, con `<U>` e `</U>` si sottolinea. Sono tutti elementi DI TESTO.

## 2.6 Nidificazione dei tag

In un documento HTML i tag sono sempre nidificati (cioè inseriti uno nel contenuto di uno precedente). Abbiamo visto ad esempio come l'elemento `body` sia contenuto all'interno dell'elemento `html`. Consideriamo ora la seguente porzione di codice HTML:

```
<P>  
<B>Esempio con tag <I>nidificati</I></B>  
</P>
```

Abbiamo il marcatore per il corsivo all'interno di quello per il grassetto che, a sua volta è all'interno di quello per la separazione dei paragrafi. L'effetto di questi tag farà visualizzare il testo in grassetto e la parola "nidificati" anche in corsivo.

Riguardo alla nidificazione ci sono alcune regole da ricordare:

- gli elementi devono essere completamente nidificati e non chiusi nell'ordine sbagliato;
- gli elementi di testo possono essere nidificati in elementi di blocco o in altri di testo;
- gli elementi di blocco possono essere nidificati solo in altri elementi di blocco.

Alla luce di queste regole i seguenti esempi risultano errati:

```
<B><I>  
    Esempio con nidificazione sbagliata per colpa dei tag di chiusura  
</B></I>  
  
<B>Esempio sbagliato con tag di <P>blocco</P> interno a uno di testo</B>
```

Alcuni browser riescono ugualmente ad interpretare sorgenti HTML contenenti errori simili a questi ma, come più volte detto in precedenza, è sempre bene rispettare le regole standard.

## 2.7 Set di caratteri

Le prime versioni di HTML, fino alla 3.2, utilizzano un set di caratteri chiamato *LATIN-1* o *ISO 8859-1*. (L'ISO è l'Organizzazione Internazionale per le Standardizzazioni e ha il compito di definire gli standard in vari ambiti a livello mondiale).

Questo set comprende i caratteri stampabili del codice ASCII, cioè quelli di valore decimale da 32 a 126, e altri caratteri numerati da 160 a 255 che includono simboli speciali e lettere straniere.

Con l'HTML 4.0 viene introdotto l'uso del set di caratteri *UNICODE* o *UCS* (*Universal Character Set*) che contiene 38.885 caratteri che fanno parte delle lingue scritte in tutto il Mondo. Essendo questo standard ancora abbastanza nuovo può capitare che un browser non gestisca i caratteri UCS ed inoltre non è detto che il computer che utilizziamo contenga il *FONT* di caratteri in grado di visualizzare determinati simboli UCS.



### 2.7.1 Entità per caratteri speciali

Torniamo allora al set di caratteri LATIN-1 e vediamo come si possono utilizzare i simboli speciali (quelli che non hanno codice ASCII compreso tra 32 e 126); si devono usare le ENTITÀ PER CARATTERI SPECIALI cioè dei codici che rappresentano tali simboli.

Le entità per caratteri speciali possono assumere due forme: con NOME o con NUMERO. Per esempio il simbolo "à" corrisponde all'entità `&agrave;` oppure `&#224;`.

Tutte le entità iniziano con "&" e finiscono con ";" , in quelle con numero le cifre sono precedute dal simbolo "#".

Di seguito sono elencati alcuni dei caratteri speciali più usati (almeno in Italia) con i rispettivi codici:

Carattere	Nome	Numero
à	<code>&amp;agrave;</code>	<code>&amp;#224;</code>
è	<code>&amp;egrave;</code>	<code>&amp;#232;</code>
é	<code>&amp;eacute;</code>	<code>&amp;#233;</code>
ì	<code>&amp;igrave;</code>	<code>&amp;#236;</code>
ò	<code>&amp;ograve;</code>	<code>&amp;#242;</code>
ù	<code>&amp;ugrave;</code>	<code>&amp;#249;</code>
£	<code>&amp;pound;</code>	<code>&amp;#163;</code>
È	<code>&amp;Egrave;</code>	<code>&amp;#200;</code>

### 2.7.2 Entità per caratteri riservati

Con lo stesso sistema viene risolto anche il problema dell'uso di caratteri "normali" ma che per HTML hanno un significato particolare come i simboli di maggiore e minore, la "&" e le virgolette. Si usano in questo caso le ENTITÀ PER CARATTERI RISERVATI con le stesse regole di sintassi:

Carattere	Nome	Numero
"		<code>&amp;#34;</code>
<	<code>&amp;lt;</code>	<code>&amp;#60;</code>
>	<code>&amp;gt;</code>	<code>&amp;#62;</code>
&	<code>&amp;amp;</code>	<code>&amp;#38;</code>

Per le virgolette esisterebbe anche l'entità con nome `&quot;` che però non fa parte dello standard riconosciuto dal W3C e quindi è meglio usare l'entità con numero `&#34;`.

### 2.7.3 Lo spazio unificatore

Tra i caratteri speciali ce n'è uno molto importante: lo SPAZIO UNIFICATORE con il quale si inserisce uno spazio tra due parole facendo in modo però che esse rimangano sempre nella stessa riga. Per inserire tale simbolo si usano le entità `&nbsp;` oppure `&#160;`.

Con lo spazio unificatore si riesce anche ad ottenere il rientro della prima riga di un paragrafo cosa impossibile da con gli spazi "normali" o i tabulatori visto che questi verrebbero compattati dal browser in fase di visualizzazione.



## 2.8 Altri elementi di uso frequente

### 2.8.1 I commenti

In un documento HTML si possono inserire commenti e annotazioni che non verranno visualizzate ma che possono essere utili per chi esamina il codice sorgente. I commenti devono essere racchiusi tra i caratteri: `<!--` e `-->`.

### 2.8.2 Uso dei colori

In un documento HTML si possono gestire i colori dello sfondo e del testo e si può definire un'immagine come sfondo usando alcuni attributi del tag `body`. Come prerequisito occorre però conoscere i codici *RGB* (*Red*, *Green*, *Blue*) dei colori: tali codici sono costituiti da tre coppie di numeri esadecimali che specificano rispettivamente la quantità di rosso, di verde e di blu presenti nel colore (combinando in vario modo questi tre colori fondamentali si ottengono infatti tutti i colori possibili). Per indicare il verde si dovrà avere il massimo di verde e niente rosso e blu e quindi il codice sarà `#00FF00`. Ogni codice esadecimale deve essere preceduto dal simbolo di cancelletto `#`. I codici dei colori più usati sono:

Colore	Codice RGB
bianco	#FFFFFF
marrone	#800000
rosso	#FF0000
verde	#00FF00
blu	#0000FF
giallo	#FFFF00
fucsia	#FF00FF
ciano	#00FFFF
blu scuro	#000080
grigio	#808080
violetto	#800080
nero	#000000

Gli attributi del tag `body` da usare sono `BGColor` per il colore dello sfondo, `Text` per il colore del testo e `Background` per usare un'immagine come sfondo.

Nel seguente esempio viene definito un documento con sfondo nero e testo bianco:

```
<HTML>
<HEAD>
<TITLE>Esempio con i colori</TITLE>
</HEAD>
<BODY BGColor="#000000" Text="#FFFFFF">
<B>Testo</B>
</BODY>
</HTML>
```

Se si vuole usare l'immagine contenuta nel file `clouds.jpg` come sfondo si scrive invece:

```
<BODY BACKGROUND="clouds.jpg">
```



Se gli attributi di body non vengono specificati la visualizzazione del documento avviene secondo lo standard del browser, di solito testo nero su sfondo bianco.

### 2.8.3 Elemento FONT

E' un elemento DI TESTO che consente di cambiare il colore, la dimensione ed il tipo dei caratteri utilizzando rispettivamente gli attributi COLOR, SIZE e FACE.

Per i colori si usano i codici esadecimali visti in precedenza, per il tipo carattere si usa il suo nome, per la dimensione ci sono sette possibilità numerate da 1, la più piccola, a 7, la più grande. Si può però anche indicare una dimensione relativa rispetto alla dimensione normale del testo. Ecco alcuni esempi:

```
<FONT SIZE="5">Esempio con dimensione 5</FONT>  
<FONT SIZE="-1">Esempio con dimensione minore di uno di quella normale</FONT>  
<FONT SIZE="+1">Esempio con dimensione maggiore di uno di quella normale</FONT>
```

Gli ultimi due esempi equivalgono ai seguenti in cui si usano i tag SMALL e BIG:

```
<SMALL>Esempio con dimensione minore di uno di quella normale</SMALL>  
<BIG>Esempio con dimensione maggiore di uno di quella normale</BIG>
```

Naturalmente i tre attributi possono essere combinati come nel seguente esempio:

```
<FONT COLOR="#00FF00" SIZE="2" FACE="Arial">Esempio con colore verde, grandezza 2  
e tipo Arial</FONT>
```

L'elemento FONT può essere utilizzato anche per modificare colore e dimensione delle intestazioni se viene nidificato all'interno dei tag <H1>, <H2> ecc.

### 2.8.4 Elemento ADDRESS

Questo elemento DI BLOCCO viene usato per inserire informazioni che riguardano l'autore del documento ed è contraddistinto dai tag <ADDRESS> e </ADDRESS>.

### 2.8.5 Elemento DIV

L'elemento DI BLOCCO DIV suddivide il documento in sezioni ed è delimitato dai tag <DIV> e </DIV>. Ogni sezione può essere definita con un particolare allineamento del testo grazie all'uso degli stessi attributi visti per paragrafi e intestazioni. Se ad esempio si vuole centrare una porzione di testo composta da molti paragrafi, invece di centrarli uno ad uno si può ricorrere al tag DIV nel seguente modo:

```
<DIV ALIGN="CENTER">
```

*testo da centrare*

```
</DIV>
```

### 2.8.6 Elemento PRE

E' un elemento DI BLOCCO che permette di visualizzare del testo nel modo in cui viene scritto nel file sorgente, senza che gli spazi siano compattati dal browser. Tutto il testo contenuto fra i tag <PRE> e </PRE> viene visualizzato così come è scritto e viene usato un carattere monospaziato (generalmente il Courier).

### 2.8.7 Elementi SUBSCRIPT e SUPERScript

Sono elementi DI TESTO che servono a trasformare il testo rispettivamente in pedice e in apice. I tag sono <SUB> e </SUB>, <SUP> e </SUP>.

Quindi per visualizzare in un documento la forma normale di un'equazione di secondo grado si deve scrivere:

$$A * X^{2} + B * X + C = 0$$



### 2.8.8 Elementi *EMPHASIS* e *STRONG*

Sono elementi DI TESTO che servono rispettivamente a enfatizzare e a enfatizzare molto quanto viene racchiuso tra i rispettivi tag `<EM>` e `</EM>`, `<STRONG>` e `</STRONG>`. Con alcuni browser questi elementi hanno lo stesso effetto rispettivamente dei tag `<I>` e `<B>`.

## 2.9 Elenchi o Liste

Gli elenchi o liste sono elementi DI BLOCCO e sono fra i più usati in HTML insieme ai paragrafi e alle intestazioni. Sono definiti cinque tipi di liste:

- liste numerate e ordinate
- liste puntate con richiamo grafico
- liste a glossario o di definizione
- liste a menu
- liste a directory

Gli ultimi due tipi sono usati molto raramente e sono anzi sconsigliati; esaminiamo quindi solo le altre tre tipologie di elenchi.

### 2.9.1 Liste numerate

I tag di inizio e fine di una lista numerata sono `<OL>` e `</OL>`; ciascuna voce dell'elenco deve essere poi preceduta dal marcatore `<LI>` che non necessita obbligatoriamente del corrispondente `</LI>`. E' possibile stabilire il tipo di numerazione desiderata con l'attributo `TYPE` del tag `<OL>`:

<code>TYPE="1"</code>	numeri arabi
<code>TYPE="a"</code>	carattere minuscolo
<code>TYPE="A"</code>	carattere maiuscolo
<code>TYPE="i"</code>	numeri romani minuscoli
<code>TYPE="I"</code>	numeri romani maiuscoli

Inoltre si può stabilire il punto di partenza della numerazione con l'attributo `START`.

Le impostazioni di default sono: numerazione araba e partenza da 1.

Anche il tag `<LI>` prevede un attributo, `VALUE` per assegnare ad una certa voce un valore specifico.

Il seguente codice:

```
<OL TYPE="I" START="1">
<LI> Penne
<LI> Matite
<LI> Quaderni
<LI VALUE="5"> Libri
<LI> Zaini
</OL>
```

viene visualizzato dal browser nel seguente modo:



1. Penne
2. Matite
3. Quaderni
5. Libri
6. Zaini

In questo esempio la lista contiene solo testo non formattato; è comunque possibile inserire qualsiasi elemento di blocco o di testo come voce dell'elenco.

### 2.9.2 Liste puntate

I tag di inizio e fine di una lista non numerata sono `<UL>` e `</UL>`; ciascuna voce dell'elenco deve essere poi preceduta dal marcatore `<LI>` che non necessita obbligatoriamente del corrispondente `</LI>`. E' possibile stabilire il tipo di carattere di richiamo degli elementi con l'attributo TYPE:

TYPE="CIRCLE"	cerchio vuoto
TYPE="DISC"	cerchio pieno
TYPE="SQUARE"	quadrato

E' importante notare che le liste (anche quelle numerate) possono essere nidificate in modo da creare delle sottoliste. I caratteri standard di richiamo sono il disco per le liste principali, il cerchio per le sottoliste di primo livello e il quadrato per le altre.

Esempio (i rientri utilizzati nel sorgente HTML hanno il solo scopo di aumentare la leggibilità dello stesso in quanto, come più volte detto, gli spazi sono ignorati dal browser):

```
<UL>
<LI>Hardware
  <UL>
    <LI> Unit&agrave; centrale
      <UL>
        <LI>CPU
        <LI>Memoria centrale
        <LI>Clock      </UL>
    <LI> Periferiche
      <UL>
        <LI>Memorie di massa
          <UL>
            <LI>Nastro
            <LI>Floppy Disk
            <LI>Hard Disk
            <LI>CD-ROM    </UL>
          <LI>Stampante
          <LI>Tastiera
          <LI>Mouse
          <LI>Schermo    </UL>
        </UL>
      <LI>Software
        <UL>
          <LI>Software di base
          <LI>Software applicativo    </UL>
        </UL>
```

Il risultato è:





- Hardware
  - Unità centrale
    - CPU
    - Memoria centrale
    - Clock
  - Periferiche
    - Memorie di massa
      - Nastro
      - Floppy Disk
      - Hard Disk
      - CD-ROM
    - Stampante
    - Tastiera
    - Mouse
    - Schermo
- Software
  - Software di base
  - Software applicativo

### 2.9.3 Liste a glossario

Le liste a glossario sono delimitate dai tag `<DL>` e `</DL>`; le voci dell'elenco sono contrassegnate dal marcatore `<DT>` e `<DD>` viene usato per fornirne la definizione.

Esempio:

```
<DL>
<DT>Hardware
<DD>Insieme degli elementi del computer che hanno una consistenza fisica
<DT>Software
<DD>Insieme dei programmi.
</DL>
```

Viene così visualizzato:

Hardware
Insieme degli elementi del computer che hanno una consistenza fisica
Software
Insieme dei programmi.



## 2.10 Collegamenti

La possibilità di definire collegamenti ipertestuali o *LINK* tra i vari documenti presenti nel WEB è senz'altro uno dei motivi del suo successo in quanto permette di passare facilmente da una pagina ad un'altra senza preoccuparsi della sua collocazione fisica. Essa può essere infatti memorizzata sulla stessa macchina o su una diversa, distante anche migliaia di chilometri (ma comunque collegata ad Internet). Inoltre è anche possibile definire dei collegamenti ad altre parti dello stesso documento.

L'entità HTML da usare per definire i collegamenti è chiamata *ancora* ed è un elemento DI TESTO. I tag da usare sono `<A>` e `</A>` e sono previsti vari attributi tra i quali i più usati sono NAME e HREF. Il testo che si trova tra i tag di apertura e chiusura (senza considerare gli attributi) è quello che il browser evidenzierà in qualche modo per attirare l'attenzione sulla presenza del link.

Ogni browser mette in risalto i link in modo diverso e di solito vengono differenziati quelli già "visitati", da quelli da visitare e da quelli attivi (un link è attivo per il breve tempo in cui viene selezionato dall'utente). NETSCAPE NAVIGATOR e INTERNET EXPLORER usano rispettivamente i seguenti colori standard: blu (#0000FF), violetto (#800080) e rosso (#FF0000).

Si possono comunque impostare i colori del testo dei collegamenti secondo i nostri gusti ricorrendo a degli attributi del tag `<BODY>` e precisamente: LINK per il colore dei collegamenti, VLINK per il colore dei collegamenti già visitati, ALINK per il colore dei collegamenti attivi.

Per esempio potremmo avere rispettivamente verde, marrone, fucsia:

```
<BODY LINK="#00FF00" VLINK="#800000" ALINK="#FF00FF">
```

### 2.10.1 Collegamenti esterni

Per collegamenti esterni si intendono quelli che puntano a documenti memorizzati su macchine diverse. Per definire questo tipo di collegamenti è necessario usare l'attributo HREF (che sta per *Hypertext REFerence*) seguito dall'URL della pagina alla quale ci si vuole collegare.

Esempio:

```
<A HREF="http://www.istruzione.it">Ministero della Pubblica Istruzione</A>
```

La scritta *Ministero della Pubblica Istruzione* viene evidenziata dal browser; la selezione di tale voce con il mouse o con la tastiera attiva il collegamento con il sito del M.P.I. ed il browser visualizza la pagina iniziale o "di benvenuto" di tale sito. Di solito è la pagina contenuta nel file index.html che viene aperta per default se, come nell'esempio in questione, non è indicato espressamente un nome di file.

Altro esempio:

```
<A HREF="http://www.rcs.it/corriere/benven.htm">Corriere della Sera</A>
```

In questo modo si definisce un collegamento con la pagina "benven.htm" del sito del Corriere della Sera.

### 2.10.2 Collegamenti a etichette in un documento

L'attributo NAME del tag `<A>` permette di inserire delle etichette che fanno riferimento a diverse sezioni di un documento. Tali etichette possono essere sfruttate quando si inseriscono, in un'altra pagina HTML, dei link al documento che le contiene: è infatti possibile collegarsi a quest'ultimo "puntando" ad una sezione specifica e non all'inizio del testo come avviene di solito.

Se ad esempio definiamo una pagina di nome "esempio.html" sulla macchina "giobix.mat.best" con delle etichette nel modo seguente:

```
.....  
.....  
<A NAME="E1">Esempio di etichetta numero 1</A>
```



.....

.....

`<A NAME="E2">Esempio di etichetta numero 2</A>`

è poi possibile collegarsi ad essa da un altro documento in vari modi come mostrato di seguito:

*Da `<A HREF="http://giobix.mat.besta/esempio.html"> qui </A>` ci si collega all'inizio della pagina esempio.*

*Da `<A HREF="http://giobix.mat.besta/esempio.html#E1"> qui </A>` ci si collega alla pagina esempio all'etichetta 1.*

*Da `<A HREF="http://giobix.mat.besta/esempio.html#E2"> qui </A>` ci si collega alla pagina esempio all'etichetta 2.*

Quindi per collegarsi ad una certa etichetta di un documento basta aggiungere al riferimento di quest'ultimo (URL e nome) il simbolo "#" seguito dal nome dell'etichetta a cui vogliamo fare riferimento.

### 2.10.3 Usare i collegamenti insieme ad altri elementi

L'elemento `<A>` può essere utilizzato insieme ad altri tag HTML ma si deve ricordare che è proibito nidificarlo in altri elementi `<A>`. Vediamo alcuni esempi:

collegamento all'interno di una intestazione

`<H3><A HREF="http://giobix.mat.besta/"> Esempio di intestazione con link</A></H3>`

si noti che `<A>` che è un elemento di testo deve essere interno al tag di intestazione che è un elemento di blocco;

collegamento in corsivo

`<I><A HREF="http://giobix.mat.besta/"> Esempio di collegamento in corsivo</A></I>`

oppure

`<A HREF="http://giobix.mat.besta/"> <I>Esempio di collegamento in corsivo</I></A>`

lista di collegamenti

`<P>I seguenti sono i server a cui ci possiamo collegare:`

`<UL>`

`<LI><A HREF="http://giobix.mat.besta/">Giobix</A>`

`<LI><A HREF="http://bella.mat.besta/">Bella</A>`

`<LI><A HREF="http://muscolis.inf.besta/">Muscolis</A>`

`<LI><A HREF="http://lazzaro.mat.besta/">Lazzaro</A>`

`<LI><A HREF="http://stella.mat.besta/">Stella</A>`

`</UL>`

Sarebbe anche possibile cambiare il colore del testo dei collegamenti, nidificando il tag `<FONT>` all'interno del tag ancora, ma questa è un'operazione sconsigliata in quanto si verrebbero a perdere le impostazioni relative ai colori di default dei collegamenti con la distinzione tra quelli visitati, non visitati e attivi e quindi l'utente non sarebbe più in grado di riconoscere i link già visitati.

### 2.10.4 Collegamenti interni

I collegamenti interni sono quelli che fanno riferimento a file residenti sul proprio sito WEB; si definiscono nello stesso modo di quelli esterni eccetto per il fatto che è possibile fare riferimento anche ad *URL relative*.



Queste ultime sono delle URL in cui l'indirizzo non è specificato interamente come avviene invece nelle *URL assolute* che abbiamo visto fino a questo momento.

Se ad esempio stiamo scrivendo una pagina HTML residente su "giobix.mat.besta/classi/4d/pippo/" e vogliamo fare riferimento al documento "esempio2.html" presente sulla stessa macchina e stessa directory, basterà scrivere:

```
<A HREF="esempio2.html">Link alla pagina esempio2</A>
```

è poi il browser che, al momento del collegamento integra l'indirizzo aggiungendo la parte mancante.

Se invece vogliamo collegarci ad un documento presente sulla stessa macchina ma in directory differenti dovremo specificare il percorso per raggiungerlo ricorrendo, se necessario, alla notazione ".." che indica la directory "madre" della directory corrente. Vediamo i seguenti esempi:

```
<A HREF="/pluto/esempio2.html">Link alla pagina esempio2</A>
```

in questo caso ci colleghiamo al file esempio2.html che è in "giobix.mat.besta/classi/4d/pippo/pluto";

```
<A HREF="../paperino/esempio2.html">Link alla pagina esempio2</A>
```

in questo caso ci colleghiamo al file esempio2.html che è in "giobix.mat.besta/classi/4d/paperino".

Per i collegamenti interni è sempre consigliato l'uso di URL relative in quanto con esse i link continuano a funzionare senza modifiche (o quasi) anche se tutti i documenti del nostro sito vengono spostati in un'altra posizione del disco o addirittura su un'altra macchina. Se si usano URL assolute questo naturalmente non è possibile e gli indirizzi dei vari collegamenti devono essere modificati in caso di spostamento dei documenti.

## 2.10.5 Collegamenti ad altre parti della stessa pagina

Grazie all'uso delle etichette è possibile definire collegamenti ad altre parti dello stesso documento in modo da poter "saltare" immediatamente ad esse. Questa possibilità può essere sfruttata in caso di pagine molto lunghe oppure per creare degli indici relativi a un documento HTML.

Ad esempio se si definisce in un documento una etichetta:

```
<A NAME="sezione1">Sezione 1</A>
```

poi si può stabilire il link ad essa scrivendo:

```
<A HREF="#sezione1">Link alla sezione 1</A>
```

Nel caso si voglia definire l'indice di un documento (di nome "esempio.html") è opportuno inserire delle etichette ad ogni intestazione del documento stesso:

```
<H1><A NAME="cap1">CAPITOLO 1</A></H1>
```

...

...

```
<H1><A NAME="cap2">CAPITOLO 2</A></H1>
```

...

ecc. ecc.

L'indice può risiedere nel documento stesso (di solito all'inizio) oppure in un altro. Nel primo caso viene definito così:

```
<A HREF="#cap1">Capitolo 1. Il sistema di elaborazione</A>
```

```
<A HREF="#cap2">Capitolo 2. I sistemi operativi</A>
```



...  
ecc. ecc.

Nel secondo caso invece (supponendo che l'indice sia in un altro documento ma nella stessa macchina e directory):

```
<A HREF="esempio.html#cap1">Capitolo 1. Il sistema di elaborazione</A>  
<A HREF="esempio.html#cap2">Capitolo 2. I sistemi operativi</A>  
...  
ecc. ecc.
```

## 2.11 Immagini nei documenti

### 2.11.1 Tipi di immagini

Nei documenti HTML è possibile inserire immagini ed i formati più comunemente utilizzati sono *GIF* (*Graphic Interchange Format*) e *JPEG* (*Joint Photographic Experts Group*). I file corrispondenti hanno rispettivamente le estensioni ".gif" e ".jpg" o ".jpeg".

Il formato GIF è stato sviluppato da Compuserve alla fine degli anni '80 ed è particolarmente adatto per linee, icone, immagini generate dal computer e con colori netti, non sfumati. Il formato JPEG si è diffuso dal 1993 ed è progettato per le fotografie e per altre immagini con colori sfumati.

Tra i due il formato GIF è senz'altro il più diffuso ed anche il meglio gestito dai vari browser. Inoltre le immagini GIF, grazie a specifici software, possono anche essere rese trasparenti e animate (qui però non approfondiamo queste possibilità). A vantaggio delle immagini JPEG c'è invece la maggiore qualità (sono "a 24 bit" cioè possono avere 16 milioni di colori contro gli "8 bit" e 256 colori delle GIF) e la compattezza: la stessa immagine in formato JPEG occupa circa un quarto dello spazio in byte del formato GIF.

La scelta tra i due formati dipende dai tipi di immagini che si vogliono utilizzare; di solito in un documento WEB si trovano entrambi.

Nel 1995 il W3C ha definito un nuovo formato di immagini per la rete, il *PNG* (*Portable Network Graphics*) che offre buona qualità e poco ingombro ma che non è ancora ben supportato dai browser più diffusi.

### 2.11.2 Inserimento di immagini

L'elemento che si utilizza per l'inserimento di immagini è `<IMG>`; è un elemento DI TESTO, prevede l'attributo `SRC` per specificare il file contenente l'immagine e l'attributo `ALT` per indicare un testo alternativo nel caso quest'ultima non possa essere visualizzata.

Esempio:

```
<IMG SRC="topolino.gif" ALT="Topolino">
```

In questo modo si inserisce l'immagine contenuta nel file di nome "topolino.gif" residente nella directory corrente; il testo alternativo è "Topolino".

I motivi per cui può essere necessario il testo alternativo sono i seguenti:

- il browser non supporta la grafica;
- il browser è programmato per pronunciare il testo al posto dell'immagine (browser per utenti ciechi);
- il browser è stato configurato solo per il testo in modo da velocizzare il caricamento dei documenti WEB;
- alcuni browser visualizzano il testo alternativo mentre l'immagine viene caricata;
- l'immagine da visualizzare non viene trovata dal browser.



### 2.11.3 Posizionamento delle immagini

Le immagini possono essere posizionate con l'attributo ALIGN del tag <IMG>. I valori possibili sono "TOP", "BOTTOM" e "MIDDLE" per l'allineamento verticale e "LEFT" e "RIGHT" per l'allineamento orizzontale.

Il valore di default è "BOTTOM" con il quale il bordo inferiore dell'immagine è allineato con la riga di testo di cui essa fa parte (si ricordi che l'elemento <IMG> è di testo).

Invece con ALIGN="TOP" è il bordo superiore dell'immagine ad essere allineato con la riga di testo e con ALIGN="MIDDLE" quest'ultimo viene allineato con la parte centrale dell'immagine.

Con le scelte ALIGN="RIGHT" e ALIGN="LEFT" l'immagine viene posizionata rispettivamente al margine destro o sinistro della pagina ed il testo che segue l'elemento <IMG> viene visualizzato affiancato all'immagine stessa. Se si vuole interrompere la visualizzazione del testo affiancato e far posizionare gli elementi successivi sotto l'immagine si deve usare l'interruzione di linea con attributo CLEAR nel seguente modo:

`<BR CLEAR="LEFT">` oppure `<BR CLEAR="RIGHT">` oppure `<BR CLEAR="ALL">`

rispettivamente nel caso che l'immagine sia sul bordo sinistro o sul bordo destro o ci siano immagini su entrambi i lati.

### 2.11.4 Dimensionamento delle immagini

Attraverso l'uso di altri attributi del tag <IMG> si possono dimensionare le immagini, impostare lo spessore del bordo e stabilire quanto spazio vuoto lasciare tra esse e il testo che le "circonda".

Gli attributi WIDTH e HEIGHT permettono di specificare rispettivamente la larghezza e l'altezza di un'immagine espressa in PIXEL.

Esempio:

`<IMG SRC="topolino.gif" WIDTH="100" HEIGHT="200" ALT="Topolino">`

L'attributo BORDER permette di specificare un bordo di un certo spessore espresso in pixel. Il default è BORDER="0" cioè nessun bordo.

Esempio:

`<IMG SRC="topolino.gif" WIDTH="100" HEIGHT="200" BORDER="10" ALT="Topolino">`

Si noti che in questo caso l'immagine avrà una larghezza totale di 120 pixel e un'altezza totale di 220 pixel.

Con gli attributi HSPACE e VSPACE si indica infine lo spazio vuoto, sempre espresso in pixel, da lasciare a sinistra o destra (con HSPACE) e sopra e sotto (con VSPACE) all'immagine.

### 2.11.5 Collegamenti con immagini

E' possibile usare immagini al posto del testo o anche insieme al testo come ancore per i collegamenti.

Esempi:

`<A HREF="http://giobix.mat.besta/"><IMG SRC="topolino.gif" ALT="Topolino"></A>`

`<A HREF="http://giobix.mat.besta/">Link a Giobix<IMG SRC="topolino.gif" ALT="Topolino"></A>`

Nel primo caso il collegamento è definito sull'immagine, nel secondo è definito sia sull'immagine che sul testo. Occorre ricordare che alcuni navigatori aggiungono automaticamente un bordo blu intorno ad una immagine collegamento per evidenziare la presenza di quest'ultimo, quindi non è opportuno definire bordi personalizzati per tali immagini..



## 3 DOCUMENTI AVANZATI

### 3.1 Informazioni supplementari nell'intestazione

La sezione HEAD di un documento HTML può contenere altre informazioni oltre al titolo del documento che, come abbiamo visto nel capitolo precedente, viene assegnato tramite il tag <TITLE>.

Esistono infatti i seguenti elementi che possono essere inseriti al suo interno:

<META>, <LINK>, <BASE>, <STYLE>, <SCRIPT>, <ISINDEX>.

#### 3.1.1 L'elemento META

E' un elemento usato per descrivere alcune proprietà del documento ed è solitamente accompagnato da due attributi: NAME o HTTP-EQUIV che a loro volta necessitano dell'attributo CONTENT.

Vediamo alcuni esempi di uso del tag META con attributo NAME:

```
<META NAME="Author" CONTENT="Paolino Paperino">  
<META NAME="Copyright" CONTENT="Walt Disney Italia">
```

in questo modo si informa il browser su chi è l'autore del documento e si danno informazioni sul copyright;;

```
<META NAME="GENERATOR" CONTENT="Mozilla/4.04 [en] (Win95; I) [Netscape]">
```

qui invece si indica il programma usato per creare il documento HTML; quando si usano editor HTML una linea simile a questa viene automaticamente inserita nella sezione HEAD. Nell'esempio l'editor usato è la versione 4.04 in inglese di NETSCAPE COMPOSER (conosciuta con il "soprannome" di Mozilla).

```
<META NAME="DESCRIPTION" CONTENT="Breve descrizione del documento">  
<META NAME="KEYWORDS" CONTENT="Qui, Quo, Qua, 313, Paperopoli, Paperina">
```

in questi esempi si danno informazioni utili per i cosiddetti *motori di ricerca*. Questi ultimi sono dei programmi in grado di indicizzare i siti WEB per poi permettere agli utenti di Internet ricerche basate sui criteri più vari. Nel caso esposto si forniscono una piccola descrizione della pagina WEB ed una serie di parole chiave riguardanti il suo contenuto, inserendo rispettivamente "DESCRIPTION" e "KEYWORDS" nell'attributo NAME e i valori corrispondenti in CONTENT.

L'attributo HTTP-EQUIV del tag META è usato per vari scopi, ad esempio:

- per causare il refresh automatico della pagina dopo un certo tempo (nell'esempio 15 secondi)

```
<META HTTP-EQUIV="REFRESH" CONTENT="15">
```

- per richiamare automaticamente un'altra pagina trascorso un certo tempo

```
<META HTTP-EQUIV="REFRESH" CONTENT="10;url=http://www.aaa.it/altrapagina.html">
```

- per evitare che il documento venga memorizzato nella cache e quindi il navigatore richieda sempre una nuova copia dello stesso

```
<META HTTP-EQUIV="EXPIRES" CONTENT="0">
```

Inoltre può essere usato per indicare la data di scadenza della pagina in modo che i navigatori prelevino la versione aggiornata al momento opportuno, oppure per classificare il contenuto del documento in modo che



possano essere attivati meccanismi di protezione per i bambini da parte dei browser. L'uso di tale attributo non viene comunque ulteriormente approfondito in questa sede.

### 3.1.2 L'elemento *LINK*

Con questo elemento si possono definire delle relazioni tra il documento ed altre pagine WEB. Non si deve confondere questo tag con un collegamento (*link* in inglese) che, come abbiamo visto, si realizza con il tag `<A>`. Sono utilizzabili i due attributi `REL` e `REV`, con il primo si stabilisce una relazione tra il nostro documento ed un'altra pagina WEB, con il secondo invece si indica che un'altra pagina è in relazione con la nostra.

Se ad esempio abbiamo una pagina chiamata "*pagina1.html*" che fa parte di un sito il cui indice è memorizzato (come avviene di solito) in "*index.html*" può essere opportuno indicare nella testata di "*pagina1.html*":

```
<LINK REL="INDEX" HREF="index.html">
```

in tal modo i navigatori e i motori di ricerca saranno informati della relazione esistente tra i due documenti e lo saranno ancor meglio se nella testata di "*index.html*" si inserisce:

```
<LINK REV="INDEX" HREF="pagina1.html">
```

Queste informazioni potrebbero anche essere sfruttate da browser avanzati che includano tra i pulsanti di navigazione un tasto "INDEX" che permetta di saltare immediatamente all'indice del sito in cui è contenuta la pagina visualizzata in un certo momento.

### 3.1.3 Gli elementi *BASE*, *SCRIPT*, *STYLE*, *ISINDEX*

Gli elementi *BASE* (con il quale si può indicare la URL in cui risiede il documento) e *ISINDEX* (indica che la pagina è in effetti un programma che effettua operazioni di ricerca) sono poco importanti ed usati molto raramente.

Invece i tag `<STYLE>`, `</STYLE>`, `<SCRIPT>`, `</SCRIPT>` sono abbastanza importanti in quanto servono rispettivamente per inserire un *FOGLIO DI STILE* (CSS, *Cascading Style Sheet*) ed uno *SCRIPT* nel documento. Tali argomenti saranno ripresi e approfonditi successivamente.

## 3.2 Uso delle tabelle

Le tabelle sono elementi molto usati per la realizzazione di documenti in HTML in quanto permettono di organizzare i dati in strutture formate da righe e colonne e di impostare il layout di una pagina disponendo i paragrafi in colonne, oppure creando dei margini, o ancora distribuendo testo ed immagini in modo più vario e movimentato.

Nelle tabelle di un documento HTML, come in qualsiasi tabella si identificano i seguenti componenti:

- un titolo, cioè una descrizione opzionale della tabella;
- le celle, cioè le intersezioni tra righe e colonne, che contengono i dati;
- le intestazioni delle righe o colonne, cioè delle celle contenenti le etichette che identificano i tipi di dati delle righe o colonne corrispondenti.

La tabella più piccola che si può creare contiene una sola cella, cioè una riga e una colonna; non ci sono invece restrizioni teoriche sul numero massimo di celle definibili. Esiste però un limite dettato da ragioni pratiche in quanto è opportuno che la tabella entri per larghezza nella finestra del browser onde evitare il ricorso, scomodo e spesso sgradito, alla barra di scorrimento orizzontale di quest'ultimo.

Le tabelle sono ormai supportate da quasi tutti i browser e ne esistono due diversi modelli: quello dell'HTML 3.2 contenente una serie molto semplice di elementi e quello dell'HTML 4.0 che ha introdotto nuovi attributi conservando però la compatibilità con il precedente.





Nella maggior parte dei casi è sufficiente conoscere il modello di tabelle più semplice; le funzioni più complesse dell'HTML 4.0 sono necessarie solo per lavori abbastanza sofisticati. Esaminiamo quindi in modo più approfondito il modello di tabelle dell'HTML 3.2.

### 3.2.1 Esempi di tabelle

Introduciamo gli elementi per la definizione delle tabelle con un semplice esempio:

```
<HTML>
<HEAD>
<TITLE>Esempio</TITLE>
</HEAD>
<BODY>
<P>
<TABLE BORDER>
<CAPTION>Tabella di prova</CAPTION>
<TR>
<TH>Alunno
<TH>Voto orale
<TH>Voto scritto
<TH>Media
</TR>
<TR>
<TD>Pippo
<TD>5
<TD>7
<TD>6
</TR>
<TR>
<TD>Pluto
<TD>6
<TD>8
<TD>7
</TR>
</TABLE>
</BODY>
</HTML>
```

L'elemento da usare per la creazione di una tabella è `<TABLE>`; è un elemento DI BLOCCO e richiede il tag `</TABLE>` alla fine della definizione della tabella stessa.

Con `<CAPTION>` e `</CAPTION>` si può inserire il titolo della tabella che comunque è opzionale.

Il tag `<TR>` definisce le varie righe suddivise nelle celle il cui contenuto è definito con `<TD>` in caso siano celle normali, o con `<TH>` in caso siano intestazioni. I tag `</TR>`, `</TD>`, `</TH>` non sono obbligatori.

L'esempio precedente viene visualizzato nel seguente modo da INTERNET EXPLORER 4:

**Tabella di prova**

Alunno	Voto orale	Voto scritto	Media
Pippo	5	7	6
Pluto	6	8	7

Vediamo anche un esempio di tabella in cui le intestazioni sono per riga invece che per colonna:



```
<HTML>
<HEAD>
  <TITLE>Esempio</TITLE>
</HEAD>
<BODY>
<P>
<TABLE BORDER CELLPADDING="9">
<CAPTION>Tabella 2</CAPTION>
<TR>
<TH>Matematica
<TD>Prof. Pippo
<TD>Ore: 5
</TR>
<TR>
<TH>Storia
<TD>Prof. Pluto
<TD>Ore: 3
</TR>
</BODY>
</HTML>
```

Il risultato in questo caso è il seguente:

**Tabella n. 2**

<b>Matematica</b>	Prof. Pippo	Ore: 5
<b>Storia</b>	Prof. Pluto	Ore: 3

Nel seguente, ulteriore esempio si ha invece una tabella con le intestazioni sia nelle righe che nelle colonne:

```
<HTML>
<HEAD>
  <TITLE>Esempio</TITLE>
</HEAD>
<BODY>
<P>
<TABLE ALIGN="CENTER" BORDER="15">
<CAPTION>Terza tabella</CAPTION>
<TR>
<TH>Alunni/Materie
<TH>Italiano
<TH>Storia
<TH>Matematica
<TH>Diritto
</TR>
<TR>
<TH>Pippo
<TD>5
<TD>7
<TD>6
<TD>6
```



```
</TR>
<TR>
<TH>Pluto
<TD>6
<TD>8
<TD>7
<TD>7
</TR>
<TR>
<TH>Paperino
<TD>6
<TD>6
<TD>7
<TD>7
</TR>
</TABLE>
</BODY>
</HTML>
```

Con questo sorgente HTML si ottiene:

**Terza tabella**

Alunni/Materie	Italiano	Storia	Matematica	Diritto
Pippo	5	7	6	6
Pluto	6	8	7	7
Paperino	6	6	7	7

### 3.2.2 Attributi dell'elemento <TABLE>

Gli attributi degli elementi per la definizione delle tabelle sono tutti opzionali; se non se ne specifica alcuno si ottiene una tabella allineata a sinistra e senza bordi.

Iniziamo a considerare gli attributi del tag <TABLE> che sono: ALIGN, BORDER, CELSPACING, CELLPADDING, BGCOLOR, WIDTH.

Con l'attributo ALIGN, usato nel terzo degli esempi precedenti, si specifica l'allineamento della tabella. I valori possono essere ALIGN="LEFT", che è il default, ALIGN="CENTER", ALIGN="RIGHT".

Attraverso BGCOLOR si specifica il colore dello sfondo della tabella usando i codici dei colori visti in precedenza.

L'attributo BORDER permette di aggiungere i bordi a tutte le celle della tabella. Se si specifica anche un valore in pixel, come nel terzo esempio, si ottiene il dimensionamento dei bordi, ma solo di quelli esterni.

Con CELSPACING="val" dove val è un valore espresso in pixel si può inserire dello spazio tra le celle che altrimenti risultano unite.

Con CELLPADDIND="val" dove val è un valore espresso in pixel si può inserire dello spazio tra il contenuto e il bordo delle celle. Questo attributo è stato usato nel secondo esempio.



Con `WIDTH` si imposta la grandezza della tabella, in percentuale rispetto alla larghezza della finestra del browser, oppure in base a un valore espresso in pixel.

Ad esempio:

```
<TABLE WIDTH="50%"> oppure  
<TABLE WIDTH="300">
```

Se questo attributo non viene usato, la tabella è dimensionata in base al contenuto della sue celle. A questo proposito è opportuno sottolineare, come emerge anche dagli esempi, che la larghezza di una colonna di una tabella è determinata dalla larghezza della cella più grande appartenente alla colonna stessa. Stessa cosa vale per l'altezza di una riga che si "adegua" all'altezza della cella più grande presente in essa.

### 3.2.3 Attributi del titolo

All'interno del titolo, cioè tra `<CAPTION>` e `</CAPTION>` si possono inserire solo elementi di testo. Con l'attributo `ALIGN` si può decidere se il titolo sta sopra (impostazione di default) o sotto la tabella:

```
<CAPTION ALIGN="TOP">Titolo sopra</CAPTION> oppure  
<CAPTION ALIGN="BOTTOM">Titolo sotto</CAPTION>
```

### 3.2.4 Attributi delle righe e delle celle

I tag `<TR>`, `<TH>`, `<TD>` prevedono gli attributi `ALIGN`, `VALIGN`, `BGCOLOR`. Esistono inoltre gli attributi `ROWSPAN`, `COLSPAN`, `WIDTH` che si applicano solo a `<TH>` e `<TD>`.

Con `ALIGN="LEFT"` o `"CENTER"` o `"RIGHT"` si imposta l'allineamento orizzontale dei dati in una cella o in tutte le celle di una riga a seconda che tale attributo sia inserito nei tag `<TD>` e `<TH>` o nel tag `<TR>`. Il valore di default è `"LEFT"`.

Discorso analogo vale per l'attributo `VALIGN` con il quale si imposta l'allineamento verticale dei dati. I valori possibili sono: `"TOP"`, `"MIDDLE"`, `"BOTTOM"`, `"BASELINE"` e il default è `"MIDDLE"`. Il valore `"BASELINE"` è simile a `"TOP"` e imposta quindi un posizionamento in alto nella cella; la differenza è nel fatto che con `"BASELINE"` la parte inferiore della prima riga di testo di ogni cella è sempre allineata qualunque sia la dimensione dei caratteri usati nelle varie celle.

L'attributo `BGCOLOR` permette di specificare il colore di sfondo di una riga o di una cella. Naturalmente l'impostazione del colore di una riga o di una cella prevale sull'impostazione dell'intera tabella. Se ad esempio abbiamo:

```
<TABLE BGCOLOR="#FF0000">  
<TR>  
<TD>Cella 1  
<TD>Cella 2  
</TR>  
<TR BGCOLOR="#00FF00">  
<TD>Cella 3  
<TD BGCOLOR="#0000FF">Cella 4  
</TR>  
</TABLE>
```

La tabella ha lo sfondo rosso ma la seconda riga lo ha verde e la cella 4, che fa parte della seconda riga, ha lo sfondo blu.

Gli attributi `COLSPAN` e `ROWSPAN` servono ad unire orizzontalmente e verticalmente più celle di una tabella. Vediamone il funzionamento con un esempio in cui si usa `ROWSPAN` (l'utilizzo di `COLSPAN` è analogo):



```
<HTML>
<HEAD>
  <TITLE>Esempio</TITLE>
</HEAD>
<BODY>
  <P>
  <TABLE BORDER>
    <CAPTION>Tabella n. 4</CAPTION>
    <TR>
      <TH ROWSPAN="2">Milano
      <TD>Minima
      <TD>-2
    </TR>
    <TR>
      <TD>Massima
      <TD>10
    </TR>
    <TR>
      <TH ROWSPAN="2">Roma
      <TD>Minima
      <TD>4
    </TR>
    <TR>
      <TD>Massima
      <TD>13
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Il risultato che si ottiene è il seguente:

**Tabella n. 4**

<b>Milano</b>	Minima	-2
	Massima	10
<b>Roma</b>	Minima	4
	Massima	13

Infine l'attributo WIDTH permette di specificare la larghezza in pixel di una cella indipendentemente dal suo contenuto. Ad esempio:

```
<TD WIDTH="20">
```

Si ricordi comunque che le celle di una colonna assumono sempre una larghezza pari a quella della cella più larga della colonna stessa.

### 3.2.5 Immagini ed altri elementi nelle celle

In una cella si può inserire praticamente tutto ciò che può essere definito nella sezione <BODY> di un documento e quindi:

- testo



- elementi di blocco (paragrafi, elenchi, altre tabelle, ecc.)
- elementi di testo (font, collegamenti, immagini, ecc.)

Grazie a questo qualche volta le tabelle vengono usate per definire il layout, cioè l'aspetto, della pagina che si sta creando. In pratica il documento viene definito come una grossa tabella ed il testo, le immagini e gli altri elementi vengono inseriti nelle sue celle. Tale utilizzo delle tabelle, pur non essendo molto ortodosso, permette di ottenere risultati estetici anche gradevoli con sforzo non eccessivo.

### 3.2.6 Creazione di celle vuote

Per creare una cella vuota è sufficiente digitare il tag `<TD>` seguito immediatamente da `</TD>`. Molti browser però non visualizzano i bordi di una cella vuota; per essere sicuri che questi siano visibili è possibile inserire nella cella vuota uno spazio unificatore:

```
<TD>&#160;</TD>
```

### 3.2.7 Tabelle nidificate

Una tabella è nidificata quando viene inserita in una cella di un'altra tabella. Tale inserimento si usa abbastanza spesso anche perché è l'unico modo per ottenere due tabelle affiancate. Vediamo proprio un esempio con due tabelle affiancate inserite in due celle di una tabella più grande definita senza bordo:

```
<HTML>
<HEAD>
  <TITLE>Esempio</TITLE>
</HEAD>
<BODY>
  <P><TABLE ALIGN="CENTER">
    <CAPTION>Tabella n.5</CAPTION>
    <TR>
      <TD>
        <TABLE BORDER="10">
          <CAPTION>I Quadrimestre</CAPTION>
          <TR><TD></TD><TH>Orale <TH>Scritto <TH>Pratico </TR>
          <TR><TD>Pippo <TD>6 <TD>7 <TD>7 </TR>
          <TR><TD>Pluto <TD>7 <TD>8 <TD>8 </TR>
        </TABLE>
      </TD>
      <TD>
        <TABLE BORDER="10">
          <CAPTION>II Quadrimestre</CAPTION>
          <TR><TD></TD><TH>Voto finale </TR>
          <TR><TD>Pippo <TD>7 </TR>
          <TR><TD>Pluto <TD>8 </TR>
        </TABLE>
      </TD>
    </TR>
  </TABLE></BODY></HTML>
```

Il risultato che si ottiene è il seguente:



Tabella n.5

I Quadrimestre				II Quadrimestre	
	Orale	Scritto	Pratico		Voto finale
Pippo	6	7	7	Pippo	7
Pluto	7	8	8	Pluto	8

### 3.3 Cenni a FRAME, CSS, Oggetti multimediali

In questo paragrafo vengono illustrati velocemente alcuni argomenti "avanzati" e relativamente "nuovi" dell'HTML. Per maggiori approfondimenti si consiglia la consultazione di manuali di HTML 4.0.

#### 3.3.1 Uso dei FRAME

I *FRAME* (riquadri) sono delle suddivisioni della finestra del browser in cui possono essere visualizzati documenti HTML diversi. In pratica con essi si possono presentare pagine WEB multiple in finestre indipendenti definite all'interno di una finestra principale del navigatore.

La prima versione di HTML che include i frame è la 4.0; non tutti i browser sono però in grado di supportarli ed inoltre i documenti che li contengono non sempre vengono visualizzati nello stesso modo. Se a questo aggiungiamo che frame non ben strutturati possono confondere gli utenti, che causano problemi di stampa, che alcuni motori di ricerca non lavorano bene in loro presenza e che spesso si crea confusione tra la URL del documento che contiene i frame e le URL dei documenti che sono in essi contenuti, si può concludere che il loro uso è raccomandabile solo quando veramente necessario.

Ad esempio possono essere utili se si hanno siti WEB con molti livelli di pagine; in tal caso si potrebbe pensare di usare un frame per visualizzare un indice del sito in modo permanente ed un altro per contenere le varie pagine che si richiamano durante la navigazione.

Per definire una pagina che deve contenere frame si usa un documento HTML particolare chiamato *FRAMESET* che si differenzia da quelli visti finora in quanto contiene l'elemento `<FRAMESET>` al posto di `<BODY>`.

Vediamo un esempio:

```
<HTML>
<HEAD>
<TITLE>Esempio con i FRAME</TITLE>
</HEAD>
<FRAMESET COLS="50%,50%" BORDERCOLOR="#00FF00">
  <FRAME SRC="doc1.html">
  <FRAME SRC="doc2.html" NAME="FRAME2">
</FRAMESET>
<BODY>
```

*Se vedete questo messaggio il vostro browser non gestisce i frame.*

*Per visualizzare i documenti cliccare sui link:*

`<A HREF="doc1.html">Primo documento</A>`

`<A HREF="doc2.html">Secondo documento</A>`



```
</BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```

Il bordo del riquadro viene impostato di colore verde con l'attributo BORDERCOLOR (il default è grigio). In questa pagina si avrà una suddivisione in due colonne uguali, grazie all'attributo COLS="50%,50%", ed in esse sono contenuti rispettivamente i documenti doc1.html e doc2.html. Se si vogliono definire frame in orizzontale si usa ROWS al posto di COLS e si possono anche definire delle "griglie" di frame unendo i due attributi; ad esempio:

```
<FRAMESET COLS="300,*" ROWS="25%,50%,25%">
```

La grandezza dei riquadri può essere specificata anche in pixel come si vede nell'esempio in cui la prima colonna è grande 300 pixel e la seconda colonna occupa lo spazio restante nella finestra (\*).

E' anche possibile usare frame nidificati continuando a dividere un riquadro in più parti fino ad un massimo di nove frame presenti nella finestra. Si deve però tenere presente che l'uso contemporaneo di più di 2-3 riquadri può favorire errori nella realizzazione del documento e può creare disorientamento nell'utente finale che lo visualizza.

Quando si usano i riquadri nidificati, il primo frameset si chiama frameset di appartenenza e gli altri frameset dipendenti.

Esempio con frameset nidificati:

```
<HTML>
<HEAD>
<TITLE>Esempio con i FRAME nidificati</TITLE>
</HEAD>
<FRAMESET COLS="200,*" BORDERCOLOR="#00FF00">
  <FRAME SRC="doc1.html">
  <FRAMESET ROWS="50%,50%">
    <FRAME SRC="doc2.html" NAME="FRAME2">
    <FRAME SRC="doc3.html" NAME="FRAME3">
  </FRAMESET>
</FRAMESET>
</HTML>
```

In questo modo si ottiene una finestra suddivisa in due colonne, una larga 200 pixel e l'altra la restante parte dello spazio e a sua volta suddivisa in due righe di uguale grandezza.

La parte compresa tra <NOFRAMES> e </NOFRAMES>, nell'esempio precedente, viene utilizzata dal browser in caso esso non sia in grado di gestire i frame; infatti rispecchia la struttura di un documento HTML "normale" con i tag <BODY> e </BODY>.

L'attributo NAME del tag <FRAME> serve ad assegnargli un nome. Ciò è utile nel caso si voglia fare in modo che i collegamenti attivati su un frame abbiano "effetto" sull'altro.

Ad esempio potremmo avere in doc1.html il link seguente in cui compare l'ulteriore attributo TARGET:

```
<A HREF="doc3.html" TARGET="FRAME2">Cliccare per vedere documento 3 </A>
```

Se viene attivato questo collegamento, il frame che lo contiene non si altera mentre sul frame chiamato "FRAME2" si ha la visualizzazione della pagina doc3.html.

Nel caso in un documento ci siano molti link tutti con stesso TARGET si può evitare di scrivere l'attributo per ognuno di essi ricorrendo all'elemento BASE da inserire nella sezione <HEAD> nel seguente modo:





```
<HEAD>
<TITLE>Titolo</TITLE>
<BASE TARGET="nome del frame">
</HEAD>
```

Esistono quattro nomi di destinazioni "speciali" che è possibile assegnare all'attributo TARGET e precisamente:

"\_blank" indica di caricare il documento aprendo una nuova finestra del navigatore;

"\_parent" fa riferimento al frameset di appartenenza (può essere utile se si usano riquadri nidificati);

"\_self" indica di caricare il documento nel frame corrente (questa è l'impostazione di default e quindi ha senso usare questa opzione solo nel caso si voglia cambiare la destinazione stabilita con un BASE TARGET precedente);

"\_top" è forse l'opzione più utile e permette di rimuovere tutti i riquadri presenti nella finestra.

Altri attributi di FRAME sono:

FRAMEBORDER="0" oppure "1" (il default è "1"), con "0" si elimina il bordo di separazione tra un frame e quello adiacente. In Internet Explorer e Netscape Navigator si possono usare anche i valori "NO" per avere un bordo normale e "YES" per averlo tridimensionale e si può anche impostare lo spessore del bordo con l'attributo BORDER del tag FRAMESET; questi attributi comunque sono estensioni HTML fuori dallo standard.

BORDERWIDTH, BORDERHEIGHT si possono usare per stabilire quanto spazio in pixel deve essere inserito tra i bordi del riquadro (rispettivamente i bordi destro e sinistro oppure superiore e inferiore) ed il suo contenuto.

NORESIZE permette di impedire il ridimensionamento del frame da parte dell'utente.

SCROLLING= "YES" oppure "NO" oppure "AUTO", il default è "AUTO" ed indica che le barre di scorrimento vengono aggiunte al riquadro automaticamente quando necessario; il valore "YES" fa inserire le barre di scorrimento sempre ed il valore "NO" mai (sconsigliato).

### 3.3.2 L'elemento *INLINE FRAME*

L'elemento *INLINE FRAME* è un elemento di TESTO introdotto con l'HTML 4.0 e permette di visualizzare un documento separato come parte di una pagina. I tag da usare sono *<IFRAME>* e *</IFRAME>* con gli stessi attributi del tag FRAME ed in più WIDTH, HEIGHT e ALIGN con ovvio significato. Un inline frame dovrebbe sempre contenere anche del testo alternativo da visualizzare se il browser non supporta tale elemento.

Vediamo un esempio:

```
<HTML>
<HEAD>
<TITLE>Esempio con i IFRAME</TITLE>
</HEAD>
<BODY>
Dopo questo testo abbiamo un elemento IFRAME contenente un altro documento HTML
<IFRAME SRC="doc1.html" ALIGN="CENTER" BORDERCOLOR="#00FF00"
NAME="RIQUADRO">
    <P> Testo alternativo: il browser non supporta gli IFRAME
    <P><A HREF="doc1.html">Cliccare qui per il doc.</A>
</IFRAME>
```



&lt;/BODY&gt;

&lt;/HTML&gt;

Come si vede dall'esempio anche ai riquadri IFRAME può essere assegnato un nome esattamente allo stesso scopo visto per i frame: in tal modo è possibile definire in un'altra pagina un collegamento che apra un documento nel riquadro voluto tramite l'attributo TARGET (con valore "RIQUADRO").

### 3.3.3 Formattazione con i CSS

I CSS (*Cascading Style Sheet*) o fogli di stile sono raccolte di definizioni che agiscono sull'aspetto di un documento HTML senza influire sulla sua struttura interna. In teoria i browser grafici potrebbero utilizzare qualsiasi tipo di foglio di stile, ma per ora l'unico riconosciuto e supportato è quello di livello 1 chiamato CSS1 e sviluppato dal W3C

Il vantaggio dell'uso del CSS1 è nel fatto che con esso si può intervenire sull'aspetto del documento senza interferire sul suo contenuto. Per questo in certi casi è preferibile usare i fogli di stile e non gli elementi HTML che servono allo stesso scopo (ad esempio i tag <HR>, <B>, <FONT>); ad esempio si può creare un unico foglio di stile da utilizzare per tutte le pagine di un certo sito in modo che il loro aspetto sia uniforme e che le modifiche riguardanti l'aspetto esteriore non debbano essere fatte sui singoli documenti ma solo sul foglio di stile.

Purtroppo però, a fronte di questi vantaggi, i fogli di stile presentano anche degli inconvenienti dovuti in parte alla loro natura ed in parte al fatto che sono strumenti relativamente nuovi e non ancora gestiti in modo completamente affidabile dai browser grafici. Tra i limiti più evidenti del CSS1 possiamo citare:

- non è stato pensato per definire le operazioni di layout di pagina (cosa invece possibile ai programmi di *Desktop Publishing*);
- non consente il posizionamento assoluto degli oggetti all'interno della pagina;
- neanche usando il CSS1 è possibile sapere con certezza quali font usa un certo computer e che dimensioni di finestra verranno usate per visualizzare il documento e quindi rimane il problema della mancanza di uniformità dell'aspetto di una pagina WEB visualizzata su piattaforme diverse.

Vediamo brevemente in che modo si definiscono e si usano i fogli di stile.

Prima di tutto si deve impostare il linguaggio di default (sempre text/css che corrisponde al CSS1) del foglio di stile con:

`TYPE="text/css"` come attributo di tutti gli elementi di tipo stile usati

oppure inserendo nella sezione HEAD il seguente elemento:

`<META HTTP-EQUIV="Content-Style-Type" CONTENT="text/css">`

il secondo metodo ha il vantaggio di effettuare la definizione una volta per tutte all'interno della pagina.

Successivamente si può associare un foglio di stile al documento HTML nelle due maniere di seguito illustrate.

#### 3.3.3.1 Fogli di stile incorporati

In questo caso i fogli di stile sono definiti all'interno della pagina con l'uso del tag <STYLE>; esempio:

&lt;HTML&gt;

&lt;HEAD&gt;

&lt;TITLE&gt;Esempio con foglio di stile

&lt;/TITLE&gt;

&lt;STYLE TYPE="text/css"&gt;

`H1, H2 { border: thick solid red;``text-align: center }``<! bordo spesso rosso centrato !>`



```
BODY { color: black; background: white; }
P.pippo { font-size: larger;
          font-weight: bolder;
          text-align: center; }

</STYLE>
</HEAD>
<BODY>
<H1> TITOLO di tipo H1 </H1>
<P>
Questo documento &egrave; realizzato con un foglio di stile incorporato
<P>
<P CLASS="pippo"> Questo paragrafo &egrave; definito nel CSS come paragrafo di tipo pippo
con font grassetto, testo centrato e di dimensioni grandi
<HR>
Questo invece &egrave; un paragrafo normale.
</BODY>
</HTML>
```

Alcune brevi spiegazioni: il foglio di stile è definito tra i tag <STYLE> e </STYLE> in esso ci sono una serie di definizioni di elementi (H1 e H2 raggruppate, BODY e P.pippo singole) con uno o più selettori a sinistra e delle dichiarazioni racchiuse tra parentesi graffe e separate da ";". Il significato delle prime due definizioni nell'esempio è abbastanza semplice da comprendere: tutti gli elementi H1, H2 e BODY del documento assumono le caratteristiche dichiarate nel foglio di stile.

Qualche considerazione in più merita la definizione di P.pippo: con essa si definisce un paragrafo (P) di "tipo" pippo con determinate caratteristiche; successivamente nel documento attraverso l'uso dell'attributo CLASS si fa riferimento a tale tipo di paragrafo.

Il risultato che si ottiene con il sorgente dell'esempio è il seguente:



Per maggiori dettagli sulla sintassi del linguaggio CSS1 si rimanda ai testi dedicati all'argomento.

### 3.3.3.2 Fogli di stile esterni

Il modo migliore per associare un foglio di stile ad una pagina HTML è quello di definirlo esternamente alla pagina stessa (come detto in precedenza questo permette di definire uno stile standard anche per un intero sito WEB e di concentrare le modifiche solo sul foglio di stile e non su tutte le pagine).

Per il collegamento al foglio esterno si usa il tag <LINK> nella sezione HEAD con gli attributi HREF e REL. Vediamo un esempio:

```
<HTML>
<HEAD>
<TITLE> Esempio con foglio di stile esterno </TITLE>
```



```
<META HTTP-EQUIV="Content-Style-Type" CONTENT="text/css">
<LINK HREF="foglio1.css" REL="STYLESHEET">
</HEAD>
```

L'attributo REL può avere anche il valore "ALTERNATE STYLESHEET" per definire un foglio di stile alternativo; inoltre si può aggiungere l'attributo TITLE per indicare il nome del foglio (questo servirà in futuro quando i browser saranno in grado di mostrare all'utente la lista dei fogli di stile associati ad un documento permettendo di disabilitare quelli non desiderati).

Il foglio di stile a cui si riferisce l'esempio si chiama foglio1.css e potrebbe avere il seguente contenuto (definito con un normale editor di testo):

```
/* Esempio di foglio di stile */
@import url(http://lazzaro.mat.besta/libcss/foglio2.css)
H1, H2 { border: thick solid red;
          text-align: center }
BODY { color: black; background: white; }
P.pippo { font-size: larger;
           font-weight: bolder;
           text-align: center; }
```

La riga racchiusa tra /\* e \*/ è un commento; la riga @import serve ad importare all'interno del foglio di stile ulteriori definizioni contenute nel foglio a cui fa riferimento l'url (in questo caso foglio2.css nella directory libcss del server WEB lazzaro.mat.besta; le altre definizioni sono le stesse utilizzate anche nell'esempio con il CSS incorporato).

### 3.3.4 Oggetti multimediali

Per inserire oggetti multimediali (suoni e filmati) in un documento HTML si usa principalmente l'elemento di TESTO <OBJECT> che però è una novità dell'HTML 4.0 e quindi può non essere supportato correttamente da tutti i browser. Questi, per riprodurre suoni e filmati, usano dei programmi chiamati *PLUG-IN* che sono quasi sempre incorporati al loro interno.

Grazie alla tecnica dello *streaming* è anche possibile ascoltare suoni e vedere filmati in tempo reale (cioè durante il trasferimento, senza dover attendere che il file relativo all'oggetto multimediale sia stato completamente scaricato dalla rete) ma sono necessari appositi programmi come REALAUDIO o REALVIDEO della società NetStream. Si deve comunque tenere conto che la tecnologia attuale (seppure in costante evoluzione) permette ai personal computer di riprodurre filmati solo in una porzione ridotta dello schermo e ai "normali" utenti di Internet, connessi via modem, di ricevere i dati relativi ai video ad una velocità che provoca talvolta una riproduzione "a scatti".

I formati più comuni dei file audio sono:

.au	Unix audio
.wav	Microsoft Wave
.aif	Macintosh audio
.ra o .ram	RealAudio
.mid o midi	MIDI ( <i>Music Instrument Digital Interface</i> )
.MP3	standard MPEG-1 audio layer III

I primi quattro contengono suoni digitalizzati con una qualità non eccezionale e una richiesta di memoria notevole per pochi secondi di registrazione. I file MIDI invece contengono "istruzioni musicali" che vengono interpretate ed eseguite dalla scheda audio del computer che così può riprodurre il brano. I file midi sono molto piccoli e offrono suoni di elevata qualità (almeno se si ha una scheda audio di discreta qualità dotata di wave table).

I file MP3 sono sicuramente i più "famosi" e stanno cambiando la maniera di acquisire, ascoltare, archiviare musica. L'MP3 è uno standard di compressione di musica digitalizzata che permette di memorizzare brani con qualità pari a quella dei CD occupando circa un decimo dello spazio che sarebbe necessario senza la



compressione (una canzone di tre minuti occupa circa 3 MB di memoria). Diventa quindi possibile scaricare i brani da internet anche con una normale connessione casalinga, ascoltarli con opportuni lettori software e conservarli sul proprio hard disk. Già si trovano in commercio lettori MP3 simili a walk-man ma privi di qualsiasi supporto magnetico o ottico in quanto contengono i brani in una memoria interna aggiornabile. Non è qui il caso di esaminare le questioni legali che la diffusione dei file MP3 in internet ha scatenato con la contrapposizione tra le maggiori case discografiche ed alcuni siti che permettevano di scaricare gratuitamente brani musicali. Sicuramente possiamo prevedere che l'MP3 sarà per i CD quello che i CD sono stati per i dischi in vinile: il futuro della diffusione della musica sarà basato su Internet e sul formato MP3 o sulle sue evoluzioni.

Passando ai formati video abbiamo i seguenti tra i più diffusi:

.mpg o .mpeg	MPEG ( <i>Moving Pictures Expert Group</i> )
.avi	Microsoft AVI ( <i>Audio Video Interleave</i> )
.mov o .qt	QuickTime

Tutti i tipi di file video supportano anche l'audio associato ma occupano una grande quantità di memoria (almeno 1 Mb per 30 secondi di filmato). I migliori da questo punto di vista sono i filmati MPEG che offrono anche una buona qualità di riproduzione.

### 3.3.4.1 *Attributi degli oggetti multimediali*

L'elemento <OBJECT> prevede gli attributi DATA, TYPE e molti degli attributi già visti per le immagini (per definire larghezza, altezza, allineamento, bordo, ecc.). Non è però previsto l'attributo ALT in quanto il testo alternativo si inserisce semplicemente tra i tag <OBJECT> e </OBJECT>.

Esempi:

```
<OBJECT DATA="symphony.mid" TYPE="audio/midi">Sinfonia in formato midi</OBJECT>
```

```
<OBJECT DATA="meteo.mov" TYPE="video/quicktime" ALIGN="LEFT" WIDTH="200"
HEIGHT="150">Filmato da satellite meteo</OBJECT>
```

L'attributo TYPE può essere tralasciato ma si deve essere sicuri che il browser sarà poi in grado di identificare il tipo di oggetto multimediale solo dall'estensione del file in questione.

In certi casi è possibile specificare altri tag: ad esempio, nel caso di file midi, per far partire automaticamente la riproduzione del brano:

```
<OBJECT DATA="symphony.mid" TYPE="audio/midi">
<PARAM NAME="AUTOSTART" VALUE="TRUE">
Sinfonia in formato midi
</OBJECT>
```

### 3.3.5 *Estensioni HTML*

Le estensioni sono degli elementi dell'HTML definiti al di fuori dello standard ufficiale ma spesso utilizzati dai browser più diffusi. Naturalmente il loro utilizzo non è consigliato se non si è sicuri che tutti coloro che navigano nelle nostre pagine WEB possono usufruire delle caratteristiche proprie di tali estensioni. In ogni caso il loro utilizzo non provoca danni: se non sono supportate sono quasi sempre semplicemente ignorate dal browser.

Alcune delle estensioni più diffuse sono le seguenti:

- testo lampeggiante (funziona in Netscape Navigator):

```
<BLINK>testo che deve lampeggiare </BLINK>
```

- testo scorrevole (funziona con Internet Explorer):



```
<MARQUEE ALIGN="TOP" BEHAVIOR="SCROLL" DIRECTION="LEFT"  
  BGCOLOR="#FF0000" LOOP="INFINITE" SCROLLDELAY="100"  
  HEIGHT="200" WIDTH="200">
```

*Testo da far scorrere*

```
</MARQUEE>
```

altri valori possibili per BEHAVIOR sono "SLIDE" e "ALTERNATE";  
HEIGHT e WIDTH permettono di dimensionare l'area del testo;  
SCROLLDELAY dà il ritardo (in millisecondi) tra scorrimenti successivi.

- linee colorate (funziona con Internet Explorer):

```
<HR COLOR="#FF0000" SIZE="5">
```

- musica di sottofondo (funziona con Internet Explorer)

```
<BGSOUND SRC="canon.mid" LOOP="INFINITE">
```

si noti che per avere la musica di sottofondo si può anche ricorrere al seguente elemento da definire nella sezione HEAD (questa non è una estensione):

```
<META HTTP-EQUIV="REFRESH" CONTENT="5; URL=canon.mid">
```

(il 5 significa che la riproduzione parte dopo 5 secondi)

- oggetti multimediali (funziona con IE e NV)

```
<EMBED SRC="canon.mid">
```

*Cliccare sul pulsante PLAY per ascoltare il brano*

```
<NOEMBED>
```

*Il browser non è in grado di riprodurre questo oggetto*

```
</NOEMBED>
```

```
</EMBED>
```

Il testo compreso tra <NOEMBED> e </NOEMBED> viene visualizzato se il browser non è in grado di riprodurre l'oggetto incorporato.

- Immagini di sfondo in una tabella (funziona con IE e NV):

```
<TABLE>
```

```
<TR>
```

```
<TD BACKGROUND="immagine.gif">Cella con sfondo</TD>
```

```
<TD>Cella senza sfondo</TD>
```

```
</TR>
```

```
</TABLE>
```

### 3.3.6 Immagini mappate

Abbiamo visto in precedenza come sia possibile inserire in un documento una immagine che agisca come link. Solitamente però l'effetto dell'attivazione del collegamento è sempre lo stesso in qualsiasi punto dell'immagine lo si attivi. Con le immagini mappate è invece possibile definire collegamenti diversificati per le varie zone di una immagine.

I tag da utilizzare per includere delle immagini a mappa in un documento sono <MAP> e <AREA> nel seguente modo:

```
<MAP NAME="nomemappa">
```



```
<AREA SHAPE="RECT" COORDS="x1,y1,x2,y2" HREF="doc1.html">
<AREA SHAPE="RECT" COORDS="w1,w2,z1,z2" HREF="doc2.html">
....
....
</MAP>
```

Nell'esempio viene definita una immagine di mappa con nome "nomemappa" e varie aree definite come rettangoli con vertici x1,y1 x2,y2 w1,w2 z1,z2 ecc. che puntano rispettivamente a doc1.html, doc2.html ecc.

Altri valori possibili di SHAPE sono:

"CIRCLE" in tal caso l'area è un cerchio e con COORDS="x,y,raggio" si indicano le coordinate del centro e la misura del raggio;

"POLY" in tal caso l'area è un poligono e con COORDS="x1,y1,x2,y2,...,xN,yN" si indicano i vertici del poligono.

Per usare la mappa così definita si deve poi ricorrere al tag IMG con un l'attributo USEMAP:

```
<IMG SRC="immag.jpg" USEMAP="#nomemappa">
```

si deve notare che il valore si USEMAP è un indirizzo URL standard; nell'esempio si fa riferimento ad una mappa definita nello stesso documento in cui è inserito l'elemento IMG, nulla vieta di usare mappe definite in file esterni o addirittura in altri siti WEB.



## 4 PAGINE INTERATTIVE E DINAMICHE

### 4.1 Introduzione

Fino a questo momento ci siamo occupati della creazione di pagine WEB "statiche" cioè di documenti contenenti testo e grafica consultabili senza possibilità di interazione da parte dell'utente; grazie però all'esistenza dell'interfaccia CGI (*Common Gateway Interface*) è possibile che un server WEB fornisca anche contenuti dinamici.

Prima di tutto occorre chiarire il significato di tale termine: una pagina dinamica non è una pagina contenente oggetti in movimento, bensì un documento contenente oggetti, dati, informazioni che possono variare anche in base all'interazione dell'utente con il documento stesso. Un esempio classico è quello di una persona che richiede su Internet informazioni su un prodotto inserendo i parametri di ricerca in una pagina WEB definita a tale scopo; i dati vengono inviati al server WEB (presumibilmente della ditta che commercializza il prodotto) che interroga il database aziendale e costruisce la pagina HTML di risposta da reinviare al browser dell'utente.

L'interfaccia CGI permette appunto ad un server WEB di ricevere dati di input da un client ed eseguire su di essi elaborazioni il cui output viene poi passato al browser in esecuzione sul client.

Ulteriori possibilità di definire pagine WEB dinamiche si hanno grazie all'uso delle APPLET o degli SCRIPT ATTIVI definiti con linguaggi di scripting come JAVASCRIPT o PHP.

Vediamo una prima classificazione di questi strumenti in base alle loro caratteristiche:

- una applicazione CGI può essere scritta con uno qualsiasi dei linguaggi di programmazione esistenti, anche quelli "classici" come C o FORTRAN, è "server-side" in quanto viene eseguita sulla macchina in cui risiede il server WEB ed è "esterna", cioè fisicamente scritta in un file separato da quello contenente il codice della pagina HTML;
- le APPLET e gli SCRIPT ATTIVI sono "client-side" cioè eseguiti sulla macchina in cui risiede il programma di navigazione, le prime sono esterne i secondi invece sono "embedded" in quanto il loro codice è scritto all'interno dello stesso file che contiene il sorgente HTML;
- gli script PHP, infine, sono server-side ed embedded.

Ancho il linguaggio HTML mette a disposizione degli elementi necessari alla definizione di documenti dinamici: i FORM o MODULI, grazie ai quali possono essere raccolti i dati immessi dagli utenti.

Inizieremo l'esame delle pagine interattive proprio dai form illustrandone l'uso in associazione ai programmi o script CGI. Successivamente verranno trattati: l'interfacciamento ai database con PHP, gli SCRIPT ATTIVI, le APPLET e verranno fatti cenni al DYNAMIC HTML, alle SERVLET, al WEB ad OGGETTI. Non saranno invece prese in considerazione soluzioni proprietarie quali ad esempio le ASP (*Active Server Page*).

### 4.2 Moduli e programmi CGI

I moduli si definiscono in un documento HTML con il tag <FORM> ma la loro gestione, o meglio, la gestione dei dati in essi contenuti, non avviene con gli strumenti dell'HTML ma grazie a dei programmi esterni chiamati programmi o script CGI.

Occorre quindi esaminare due aspetti distinti:

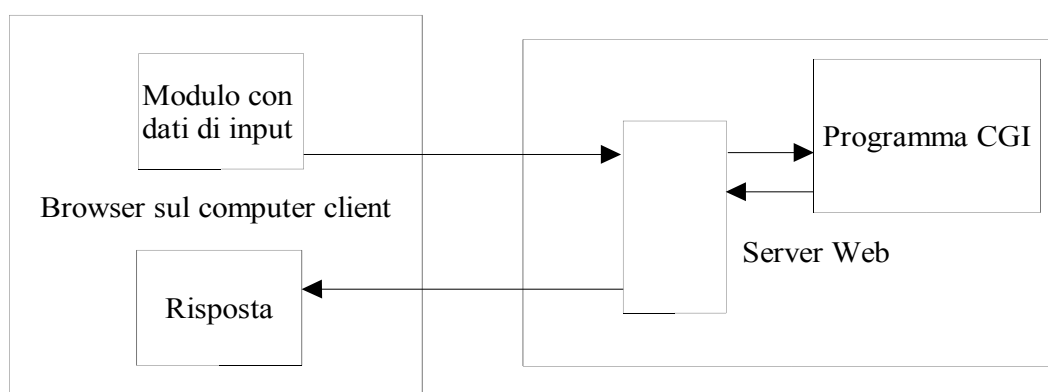
- definizione in HTML degli elementi che compongono il modulo;
- scrittura del programma, in linguaggio di programmazione, che deve gestire i dati immessi con esso.





Scendendo più in dettaglio, la corretta gestione di un form passa attraverso le seguenti fasi:

- inserimento nella pagina HTML dei tag per la definizione del modulo;
- indicazione, tramite un attributo dell'elemento FORM, del nome del programma esterno che deve elaborare i dati;
- visualizzazione del form da parte del browser che raccoglie gli input dell'utente e li invia al server WEB;
- attivazione, da parte del server WEB, del programma indicato al quale sono forniti gli input ricevuti dal modulo;
- elaborazione dei dati da parte del programma
- restituzione dei risultati al browser, spesso mediante la creazione di una nuova pagina WEB da visualizzare.



I CGI sono di due tipi in base al linguaggio di programmazione utilizzato per scriverli:

PROGRAMMI CGI, se si usano linguaggi compilati come C, C++, Java;

SCRIPT CGI, se si usano linguaggi interpretati come Perl, script di shell UNIX o Linux.

Nel proseguo useremo il termine programmi CGI o anche solo CGI per riferirci ad entrambe le tipologie.

Di solito tutti i CGI vengono memorizzati in una stessa directory del server WEB di nome cgi-bin.

Esiste comunque un'alternativa per la gestione dei dati di un modulo che consiste nell'inviarli ad un indirizzo di posta elettronica anziché ad un programma esterno.

Nei prossimi due paragrafi si prenderanno in esame le due fasi della gestione di un form: la sua definizione in HTML e l'elaborazione dei dati con i programmi CGI.

### 4.2.1 I moduli

I form (moduli) iniziano con il tag `<FORM>` che richiede la chiusura `</FORM>`; è un elemento di BLOCCO e può contenere qualsiasi altro elemento ma non altri form; i suoi attributi sono:



ACTION, per specificare l'URL a cui inviare i dati del form;  
METHOD, per specificare l'azione da svolgere sui dati, i valori possibili sono "GET" o "POST";  
ENCTYPE, per indicare il tipo di MIME usato per il trasferimento delle informazioni.  
TARGET, per definire una nuova finestra dove visualizzare la risposta elaborata dal CGI.

Esistono poi i cosiddetti attributi di evento come ONMOUSEOVER, ONCLICK, ONSUBMIT, ONRESET che verranno presi in esame al momento di trattare gli script attivi e le applet.

Gli attributi più importanti sono ACTION e METHOD. Il valore di ACTION corrisponde all'indirizzo a cui inviare i dati; può essere o un indirizzo di posta elettronica o il nome del programma CGI (completo di percorso) che dovrà ricevere ed elaborare in qualche modo i dati del form.

Il valore di METHOD può essere "GET" oppure "POST"; nel primo caso i dati vengono accodati all'URL indicato in ACTION, nel secondo caso i dati vengono passati in modo autonomo attraverso lo *standard input* al server.

Torneremo più diffusamente su questi aspetti nel paragrafo dedicato alla definizione dei programmi CGI.

Riguardo all'attributo ENCTYPE è opportuno soffermarsi brevemente sui tipi MIME (*Multipurpose Internet Mail Extension*) creati originariamente per descrivere gli allegati di posta elettronica e ora usati, più generalmente, per riconoscere la natura dei file presenti sul WEB. Sono composti da due parti, tipo principale / tipo specifico come ad esempio: "image/gif" (immagini di tipo gif), "text/plain" (testo puro) e molti altri.

Un elenco abbastanza completo è il seguente:

<i>MIME type</i>	<i>Estensione del file</i>
application/activemessage	
application/andrew-inset	
application/applefile	
application/atomicmail	
application/dca-rft	
application/dec-dx	
application/mac-binhex40	hqx
application/mac-compactpro	cpt
application/macwriteii	
application/msword	doc
application/news-message-id	
application/news-transmission	
application/octet-stream	bin dms lha lzh exe class
application/oda	oda
application/pdf	pdf
application/postscript	ai eps ps
application/powerpoint	ppt
application/remote-printing	
application/rtf	rtf
application/slate	
application/wita	
application/wordperfect5.1	
application/x-bcpio	bcpio



<i>MIME type</i>	<i>Estensione del file</i>
application/x-cdlink	vcd
application/x-compress	
application/x-cpio	cpio
application/x-csh	csh
application/x-director	dcr dir dxr
application/x-dvi	dvi
application/x-gtar	gtar
application/x-gzip	
application/x-hdf	hdf
application/x-koan	skp skd skt skm
application/x-latex	latex
application/x-mif	mif
application/x-netcdf	nc cdf
application/x-sh	sh
application/x-shar	shar
application/x-stuffit	sit
application/x-sv4cpio	sv4cpio
application/x-sv4crc	sv4crc
application/x-tar	tar
application/x-tcl	tcl
application/x-tex	tex
application/x-texinfo	texinfo texi
application/x-troff	t tr roff
application/x-troff-man	man
application/x-troff-me	me
application/x-troff-ms	ms
application/x-ustar	ustar
application/x-wais-source	src
application/zip	zip
audio/basic	au snd
audio/midi	mid midi kar
audio/mpeg	mpga mp2
audio/x-aiff	aif aiff aifc
audio/x-pn-realaudio	ram
audio/x-pn-realaudio-plugin	rpm
audio/x-realaudio	ra
audio/x-wav	wav
chemical/x-pdb	pdb xyz
image/gif	gif



<i>MIME type</i>	<i>Estensione del file</i>
image/ief	ief
image/jpeg	jpeg jpg jpe
image/png	png
image/tiff	tiff tif
image/x-cmu-raster	ras
image/x-portable-anymap	pnm
image/x-portable-bitmap	pbm
image/x-portable-graymap	pgm
image/x-portable-pixmap	ppm
image/x-rgb	rgb
image/x-xbitmap	xbm
image/x-xpixmap	xpm
image/x-xwindowdump	xwd
message/external-body	
message/news	
message/partial	
message/rfc822	
multipart/alternative	
multipart/appledouble	
multipart/digest	
multipart/mixed	
multipart/parallel	
text/html	html htm
text/plain	txt
text/richtext	rtx
text/tab-separated-values	tsv
text/x-setext	etx
text/x-sgml	sgml sgm
video/mpeg	mpeg mpg mpe
video/quicktime	qt mov
video/x-msvideo	avi
video/x-sgi-movie	movie
x-conference/x-cooltalk	ice
x-world/x-vrml	wrl vrml

Il tipo di MIME da usare per trasferire i dati dal form al server HTTP è: "application/x-www-form-urlencoded" e deve essere specificato solo nel caso si usi il METHOD "POST"; questo valore è comunque quello di default dell'attributo ENCTYPE.

In pratica serve a comunicare al server che i dati saranno inviati attraverso lo standard input (il metodo è "POST") ma codificati alla stessa maniera che con il metodo "GET".



Gli elementi specifici usati all'interno dei moduli sono elementi di TESTO chiamati "controlli"; i più importanti sono:

- <INPUT> per creare vari tipi di input diversi;
- <SELECT> per creare menu a scorrimento le cui opzioni sono indicate con il tag <OPTION>;
- <TEXTAREA> per l'immissione di righe di testo multiple.
- <FIELDSET> per raggruppare più controlli di un form.

I primi tre prevedono l'attributo **NAME** che permette di indicare il nome della variabile o del campo che contraddistingue l'elemento in questione. Talvolta si utilizza anche l'attributo **VALUE** con il quale si assegna un valore alla variabile o al campo; più spesso però il valore assunto corrisponde a ciò che l'utente ha digitato o scelto in corrispondenza di quel campo.

Il nome ed il valore dei campi sono naturalmente di fondamentale importanza per il programma CGI che deve elaborare i dati inviati dal modulo e si deve prestare molta attenzione al loro diverso ruolo: in pratica siamo in presenza della distinzione tra nome di una variabile e suo contenuto, ben nota a chi conosca i rudimenti della programmazione.

Vediamo prima di tutto un esempio molto semplice di modulo con un campo di input:

```
<HTML>
<HEAD>
  <TITLE>Esempio di modulo</TITLE>
</HEAD>
<BODY>
<H4>ESEMPIO DI INPUT CON RICHIAMO DI PROGRAMMA CGI</H4>
<FORM ACTION="/cgi-bin/cgi-prova.sh" METHOD="GET">
  <P><BR><B>Inserire dato da passare al programma&nbsp;&nbsp; </B>
  <INPUT TYPE="TEXT" NAME="datoin" SIZE="5" MAXLENGTH="3">
  <INPUT TYPE="SUBMIT" VALUE="Invio">
</FORM>
</BODY>
</HTML>
```

L'aspetto del modulo è il seguente:

ESEMPIO DI INPUT CON RICHIAMO DI PROGRAMMA CGI

Inserire dato da passare al programma

Invio

In questo esempio si chiede l'inserimento di un dato che viene passato con metodo "GET" al programma "cgi-prova.sh" (presumibilmente scritto con i comandi della shell di linux o unix, data l'estensione "sh") residente nella directory "/cgi-bin". Il dato viene identificato come "datoin" dal programma, viene immesso in una casella lunga 5 caratteri ma può essere lungo al massimo 3 caratteri. L'invio dei dati avviene cliccando sul pulsante definito dal controllo "SUBMIT". In questo esempio tale controllo non ha l'attributo NAME che potrebbe invece servire nel caso si debbano distinguere vari pulsanti di invio in uno stesso modulo.

Tra gli attributi di <INPUT> il più importante è TYPE il cui valore permette di definire vari tipi di controlli:



TYPE="SUBMIT" crea un pulsante da usare per l'invio dei dati;  
TYPE="RESET" crea un pulsante per la reinizializzazione dei campi del modulo;  
TYPE="TEXT" crea un campo di testo di una sola riga;  
TYPE="RADIO" crea un pulsante di opzioni  
TYPE="CHECKBOX" crea una casella di selezione  
TYPE="IMAGE" come submit ma con una immagine al posto del pulsante  
TYPE="HIDDEN" crea un elemento che non appare nel modulo ma ha un nome e un valore  
TYPE="PASSWORD" crea un campo in cui i dati immessi sono visualizzati come asterischi  
TYPE="BUTTON" crea un pulsante generico definito dal programmatore  
TYPE="FILE" usato per inviare un file insieme ai dati del modulo

Nei prossimi due esempi sono presenti tutti i controlli della lista ancora non esaminati (ad eccezione di "BUTTON" che verrà usato in seguito) nonché altri possibili attributi del tag <INPUT>:

```
<HTML>
<HEAD><TITLE>Esempio con controlli vari</TITLE></HEAD>
<BODY>
<DIV ALIGN="CENTER"><B>Esempio con controlli vari</B></DIV>
<P><BR><P>
<FORM ACTION="/cgi-bin/cgi-prova.sh" METHOD="GET">
<INPUT TYPE="HIDDEN" NAME="nascosto" VALUE="aaa">
<P>Fascia di et&agrave;;:
<P><INPUT TYPE="RADIO" NAME="fascia" VALUE="0-15">0-15
<P><INPUT TYPE="RADIO" NAME="fascia" VALUE="16-30" CHECKED>16-30
<P><INPUT TYPE="RADIO" NAME="fascia" VALUE="30-50">30-50
<P><INPUT TYPE="RADIO" NAME="fascia" VALUE="50-99">50-99
<P><BR><P>
Seleziona i tuoi campi di interesse:
<P>
<INPUT TYPE="CHECKBOX" NAME="int" VALUE="computer" CHECKED>&nbsp;Computer
<INPUT TYPE="CHECKBOX" NAME="int" VALUE="musica">&nbsp;Musica
<INPUT TYPE="CHECKBOX" NAME="int" VALUE="sport">&nbsp;Sport
<P><BR><P>
Inserisci la password:&nbsp;<INPUT TYPE="PASSWORD" NAME="psw" SIZE="8"
MAXLENGTH="8">
<P><INPUT TYPE="IMAGE" SRC="poweredby.png" ALT="Invio" ALIGN="RIGHT">
<P><INPUT TYPE="RESET" VALUE="Pulisci campi" SIZE="10">
</FORM>
</BODY>
</HTML>
```

L'aspetto di questo modulo è il seguente:



## Esempio con controlli vari

Fascia di età:

☐ 0-15☒ 16-30☐ 30-50☐ 50-99

Seleziona i tuoi campi di interesse:

☒ Computer ☐ Musica ☐ SportInserisci la password: 

Alcuni degli attributi utilizzati in questo esempio sono di immediata comprensione, anche perché già visti in precedenza (come SRC, ALT, ALIGN). L'attributo CHECKED viene usato in caso di controlli che prevedono più alternative per preimpostarne una o alcune.

I controlli "HIDDEN" possono essere utili nel caso si debbano inviare dei dati fissi al programma CGI senza che l'utente possa vederli e tantomeno variarli.

L'invio dei dati del modulo avviene cliccando sull'immagine che appare in fondo a destra.

```
<HTML>
<HEAD><TITLE>Esempio con trasf. di file</TITLE></HEAD>
<BODY>
<DIV ALIGN="CENTER"><B>TRASFERIMENTO DI FILE</B></DIV>
<P><BR><P>
<FORM ENCTYPE="multipart/form-data" ACTION="/cgi-bin/cgi-prova.sh">
<P>
File da inviare:&nbsp;<INPUT TYPE="FILE" NAME="file">
<P>
<INPUT TYPE="SUBMIT" SIZE="5" VALUE="INVIO">
</FORM>
</BODY>
</HTML>
```

Questo modulo ha il seguente aspetto:

## TRASFERIMENTO DI FILE

File da inviare:



Il controllo "FILE" permette di inviare un file al programma CGI; presenta un bottone "BROWSE" o "SFOGLIA" inserito automaticamente per permettere la ricerca del file da inviare all'interno della macchina locale. Si noti che nel caso di invio di un file si deve usare l'attributo di <FORM> ENCTYPE con valore "multipart/form-data".

### 4.2.2.2 Elemento *SELECT*

L'elemento <SELECT> crea un menu a cascata e prevede obbligatoriamente l'attributo NAME; le opzioni del menu sono indicate ognuna con <OPTION> e al termine è necessario il tag di chiusura </SELECT>.

Altri attributi utilizzabili sono:

SIZE (attributo di <SELECT>) per indicare il numero di voci di menu visibili contemporaneamente;  
MULTIPLE (attributo di <SELECT>) per dare la possibilità di selezionare più di una voce;  
SELECTED (attributo di <OPTION>) per indicare una voce di default;  
VALUE (attributo di <OPTION>) per indicare un testo, associato alla scelta effettuata, da inviare al programma CGI; se manca tale attributo, il testo inviato corrisponde al nome dell'opzione.

### 4.2.2.3 Elemento *TEXTAREA*

Con l'elemento <TEXTAREA> si può creare una casella di testo contenente più righe. Si deve chiudere con </TEXTAREA> ed è obbligatorio l'attributo NAME. Ci sono anche gli attributi ROWS e COLS con i quali si indica l'ampiezza della casella di testo senza però limitare la quantità di caratteri inseribili. Quasi tutti i browser infatti inseriscono le barre di scorrimento a sinistra ed in basso in caso la quantità di caratteri immessi superi la dimensione della casella.

Si può anche inserire un testo di default semplicemente scrivendolo tra i tag <TEXTAREA> e </TEXTAREA>.

Nel seguente esempio viene mostrato l'uso degli elementi <SELECT> e <TEXTAREA>:

```
<HTML>
<HEAD><TITLE>Esempio con SELECT e TEXTAREA</TITLE></HEAD>
<BODY>
<DIV ALIGN="CENTER"><B>Esempio con SELECT e TEXTAREA</B></DIV>
<P><BR><P>
<FORM ACTION="/cgi-bin/cgi-prova.sh" METHOD="POST">
Componenti da acquistare:<P>
<SELECT NAME="comp" MULTIPLE SIZE="6">
<OPTION SELECTED>Main board
<OPTION SELECTED>Cpu
<OPTION SELECTED>Ram
<OPTION>Floppy disk 120 Mb
<OPTION>Floppy disk 1,44 Mb
<OPTION>Hard disk IDE
<OPTION>Hard disk SCSI
<OPTION>Scheda video
<OPTION>Scheda audio
</SELECT>
<P><BR><P>Inserire un commento<P>
<TEXTAREA NAME="commento" ROWS="5" COLS="50">
Testo inserito preventivamente
</TEXTAREA>
<P><INPUT TYPE="SUBMIT" SIZE="5" VALUE="INVIO">
</FORM>
</BODY>
</HTML>
```

Il modulo che si ottiene è il seguente:



**Esempio con SELECT e TEXTAREA**

Componenti da acquistare:

Main board

Cpu

Ram

Floppy disk 120 Mb

Floppy disk 1,44 Mb

Inserire un commento

Testo inserito preventivamente

INVIO

#### 4.2.2.4 Elemento *FIELDSET* e attributo *TABINDEX*

Con l'elemento `<FIELDSET>` si possono raggruppare, incorniciandoli automaticamente, più controlli di un form. Si raggruppano tutti i controlli presenti fino al tag di chiusura `</FIELDSET>`; all'interno si può usare il tag `<LEGEND>` per aggiungere una legenda posizionandola con l'attributo `ALIGN`.

Resta infine da segnalare l'attributo `TABINDEX`, utilizzabile per tutti i controlli di un form, con il quale si può cambiare l'ordine di selezione dei controlli all'interno di un modulo. La sintassi da usare è:

*TABINDEX="numero"*

i numeri alti hanno la precedenza rispetto a quelli bassi o negativi. I caso di assenza di questo attributo, l'ordine di selezione dei controlli corrisponde a quello di apparizione all'interno del modulo.

### 4.3 Definizione e utilizzo dei programmi CGI

#### 4.3.1 Scrittura di un CGI

Per potere scrivere ed utilizzare dei programmi CGI occorre prima di tutto che il server WEB sia configurato per la loro esecuzione e che sia possibile memorizzarli in una apposita directory dello stesso (di solito `/cgi-bin`); inoltre è necessario che sul server sia possibile eseguire i programmi nel linguaggio di programmazione scelto per realizzare i CGI (ad esempio se si scrivono in linguaggio PERL, sul server deve essere installato l'interprete PERL).

Soddisfatti questi prerequisiti si può passare alla realizzazione dei programmi. Essi generalmente ricevono dati in input dal browser tramite il server WEB (vedere il diagramma ad inizio capitolo) e possono manipolarli liberamente.

I risultati invece devono avere caratteristiche molto precise; innanzitutto devono iniziare con una particolare intestazione che non viene mai visualizzata e può essere: "Location", "Status", o "Content-type".



Quest'ultima è la più usata e serve a fornire al browser informazioni sul contenuto del documento di risposta; il suo formato è:

*Content-type: "tipo MIME"*

e deve essere seguita necessariamente da una riga vuota.

Dei tipi MIME abbiamo già parlato in precedenza, ricordiamo solo i più utilizzati in questo contesto:

Contenuto output	Tipo MIME da usare
HTML	text/html
testo	text/plain
immagine GIF	image/gif
immagine JPEG	image/jpeg
video MPEG	video/mpeg

Naturalmente il contenuto della parte rimanente dell'output deve essere congruente con quanto dichiarato nell'intestazione; quindi nel caso ad esempio Content-type sia "text/html", il programma CGI deve creare il resto del documento di risposta in linguaggio HTML.

Un altro tipo di risposta che può essere fornita da un programma CGI consiste nell'invio di una pagina WEB già pronta, residente sul server o presso qualsiasi altro indirizzo in rete. In tal caso il documento di output deve contenere solo l'intestazione che sarà di tipo "Location" con questa sintassi:

*Location: "URL della pagina da inviare"*

anche in questo caso è necessario aggiungere una riga vuota dopo l'intestazione.

Il terzo tipo di risposta si ottiene con la riga di intestazione "Status", sempre seguita da una riga vuota. Può essere utilizzata nel caso il programma CGI non debba fornire alcun risultato visibile; si scrive allora la seguente intestazione nel documento di output:

*Status: 204 No Response*

I browser sono in grado di interpretare vari codici di stato nell'ambito della connessione al server tramite il protocollo HTTP; in particolare il codice 204 viene interpretato come richiesta di non svolgere alcuna operazione.

Altri codici di risposta importanti in cui può capitare di imbattersi sono:

200 OK;  
404 Documento non trovato;  
500 Errore interno del server.

### 4.3.2 Passaggio dei dati del modulo

Al momento della definizione di un modulo, come abbiamo visto, occorre specificare il nome e la locazione del CGI che deve trattare i dati e anche il modo in cui essi vengono trasferiti al CGI stesso. Ricordando che allo scopo si usano gli attributi ACTION e METHOD del tag <FORM>, vediamo il seguente esempio di definizione di un modulo (tralasciando i dettagli dei controlli) in cui si fanno queste ipotesi:

il programma CGI si chiama cgi-prova.sh e risiede nella directory /cgi-bin del server WEB,  
nel modulo sono inseriti due campi con NAME "cognome" e "nome",  
i valori dei due campi inseriti dall'utente sono "Paperino" e "Paolino" rispettivamente.

*<FORM ACTION="/cgi-bin/cgi-prova.sh" METHOD="GET">*



```
...  
...  
</FORM>
```

In caso di invio con metodo "GET" i dati, nella forma "nome=valore", vengono accodati all'URL indicato in ACTION separandoli da esso con il carattere "?"; inoltre ogni coppia nome-valore è separata dalla successiva con il carattere "&". Quindi l'URL effettivo diventa:

"cgi-bin/cgi-prova.sh?cognome=Paperino&nome=Paolino"

Il programma cgi-prova.sh riceve i dati nella variabile di ambiente QUERY\_STRING che assume il valore "cognome=Paperino&nome=Paolino".

L'alternativa è usare il metodo "POST":

```
<FORM ACTION="/cgi-bin/cgi-prova.sh" METHOD="POST">  
...  
...  
</FORM>
```

In questo caso i dati vengono passati attraverso il canale di input standard mentre la variabile QUERY\_STRING rimane vuota; il programma CGI deve quindi preoccuparsi di leggere lo standard input dove trova la stringa: "cognome=Paperino&nome=Paolino".

Il metodo "POST" è quello da usare obbligatoriamente nel caso i dati debbano essere trasferiti ad un indirizzo di posta elettronica; negli altri casi si possono usare indifferentemente i due metodi.

Occorre però osservare che il metodo "GET" è il meno sicuro in presenza di una grossa mole di dati da trasferire in quanto essi vengono accodati all'URL e, siccome gli URL hanno lunghezza limitata, si rischia di perdere una parte dei dati inviati dal modulo al programma.

Nel caso infine si usi un modulo per trasferire un file, il programma CGI deve essere di tipo particolare: non deve aspettarsi stringhe in input né in QUERY\_STRING né nello standard input, ma deve essere predisposto alla lettura del file inviato dallo standard input.

### 4.3.3 Decodifica dell'input

Come abbiamo visto, indipendentemente dal metodo usato per l'invio, l'input arriva al programma CGI come una stringa composta da coppie nome=valore come la seguente:

"nome1=valore1&nome2=valore2&nome3=valore3.....";

questo formato di invio prende il nome di "codifica URL" e si basa sulle seguenti regole:

- ogni coppia nome=valore è separata dal carattere "&";
- gli elementi di ogni coppia sono separati dal simbolo di uguaglianza;
- se un campo non contiene alcun valore, la coppia corrispondente diventa "nome=";
- gli spazi nell'input sono rappresentati da segni "+";
- i caratteri speciali, cioè quelli con codice ASCII maggiore di 127, e i caratteri "=", "&", "%", "/", "~", "@", "+" vengono rappresentati in esadecimale preceduti dal simbolo "%".

Vediamo in dettaglio le codifiche di questi ultimi:

Carattere	-	Codifica
=	-	%3D
&	-	%26
%	-	%25



/	-	%2F
~	-	%7E
@	-	%40
+	-	%2B

Il programma CGI deve per prima cosa fare il "parsing" dell'input, cioè decodificarlo, tenendo conto di tutte le regole appena illustrate, solo successivamente può svolgere le elaborazioni sui dati così ottenuti.

Essendo la fase di decodifica necessaria in qualsiasi elaborazione CGI, esistono moltissimi strumenti, reperibili anche in Internet, che svolgono questa operazione.

Un esempio è il programma "uncgi" scritto in linguaggio C scaricabile all'indirizzo

`"http://www.midwinter.com/~koreth/uncgi.html"`,

utilizzabile su server Linux, Unix e, in certi casi anche Windows.

Una volta scaricato il programma si deve compilarlo, seguendo le istruzioni allegate, e installare l'eseguibile ottenuto nella directory apposita del server (presumibilmente /cgi-bin).

Per illustrarne il funzionamento riprendiamo in considerazione l'esempio del paragrafo precedente in cui la stringa di input era:

`"cognome=Paperino&nome=Paolino"`

per fare in modo che tale input venga decodificato dal programma uncgi l'attributo ACTION deve essere valorizzato nel modo seguente:

`ACTION="/cgi-bin/uncgi/cgi-prova.sh"`

In questo modo infatti i programmi indicati dopo "/cgi-bin" vengono eseguiti uno dopo l'altro: per primo "uncgi" che riceve la stringa di input e la decodifica.

Si deve notare che tale programma funziona indifferentemente con entrambi i metodi di invio "GET" e "POST". Al termine della sua esecuzione restituisce le variabili della stringa di input decodificate e precedute dal prefisso "WWW\_".

Il programma "cgi-prova.sh" quindi può essere scritto senza preoccuparsi della fase di decodifica e tenendo conto che le variabili provenienti dal form hanno i nomi: "WWW\_cognome" e "WWW\_nome".

### 4.3.4 Variabili di ambiente

Un programma CGI ha a disposizione un gruppo di "variabili di ambiente" che vengono valorizzate automaticamente dal server HTTP e che il programma può usare liberamente.

Segue l'elenco delle più importanti di tali variabili con una breve descrizione del loro contenuto:

NOME VARIABILE	CONTENUTO
SERVER_NAME	Nome della macchina su cui funziona il programma CGI;
SERVER_SOFTWARE	Il server WEB utilizzato, ad esempio Apache 1.2.4;
GATEWAY_INTERFACE	Versione del CGI in esecuzione (di solito CGI/1.1);
SERVER_PROTOCOL	Protocollo HTTP usato dal server (di solito HTTP/1.0);
SERVER_PORT	Porta per il collegamento al server WEB (di solito 80);
REQUEST_METHOD	Metodo usato per l'invio dei valori: "POST" o "GET";
HTTP_ACCEPT	Elenco di MIME che il browser è in grado di accettare;
SERVER_ADMIN	Indirizzo E-Mail dell'amministratore del server;
HTTP_USER_AGENT	Nome e versione del browser che ha inviato i dati;
HTTP_REFERER	Nome del documento HTML che ha inviato i dati;
SCRIPT_NAME	Nome del programma CGI completo di percorso;
QUERY_STRING	Stringa di input se il metodo di invio è "GET";
REMOTE_HOST	Nome della macchina che ha inviato i dati;
REMOTE_ADDR	Indirizzo IP della macchina che ha inviato i dati;



Questo documento, o parte di esso, può essere riprodotto e distribuito con qualunque mezzo, fisico o elettronico, purché sia accompagnato da questo copyright e da questa licenza.



```
echo
echo "SERVER_NAME = $SERVER_NAME"
echo "SERVER_SOFTWARE = $SERVER_SOFTWARE"
echo "GATEWAY_INTERFACE = $GATEWAY_INTERFACE"
echo "SERVER_PROTOCOL = $SERVER_PROTOCOL"
echo "SERVER_PORT = $SERVER_PORT"
echo "REQUEST_METHOD = $REQUEST_METHOD"
echo "HTTP_ACCEPT = $HTTP_ACCEPT"
echo "SERVER_ADMIN = $SERVER_ADMIN"
echo "HTTP_USER_AGENT = $HTTP_USER_AGENT"
echo "HTTP_REFERER = $HTTP_REFERER"
echo "SCRIPT_NAME = $SCRIPT_NAME"
echo "QUERY_STRING = $QUERY_STRING"
echo "REMOTE_HOST = $REMOTE_HOST"
echo "REMOTE_ADDR = $REMOTE_ADDR"
echo "REMOTE_USER = $REMOTE_USER"
echo "CONTENT_TYPE = $CONTENT_TYPE"
echo "CONTENT_LENGTH = $CONTENT_LENGTH"
echo "PATH_INFO = $PATH_INFO"
echo "PATH_TRANSLATED = $PATH_TRANSLATED"
echo
echo "Standard input:"
cat
echo "</PRE>"
echo "</BODY>"
echo "</HTML>"
```

Aspetto del documento di risposta creato da cgi-var.sh:

### Test variabili di ambiente CGI

```
SERVER_NAME = ferronif.inf.best
SERVER_SOFTWARE = Apache/1.3.12 (Unix) (Red Hat/Linux)
PHP/3.0.15 mod_perl/1.21
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = GET
HTTP_ACCEPT = image/png, image/*, */*
SERVER_ADMIN = root@localhost
HTTP_USER_AGENT = Mozilla/3.0 (compatible; StarOffice/5.1;
Linux)
HTTP_REFERER = http://127.0.0.1/ese4.html
SCRIPT_NAME = /cgi-bin/uncgi/cgi-var.sh
QUERY_STRING = dato=pro
REMOTE_HOST =
REMOTE_ADDR = 127.0.0.1
REMOTE_USER =
CONTENT_TYPE =
CONTENT_LENGTH =
PATH_INFO = /altri_dati/passati/via_url
PATH_TRANSLATED = /home/httpd/html/altri_dati/passati/via_url

Standard input:
```



A questo punto possiamo illustrare un esempio completo comprendente un modulo per l'immissione di alcuni campi di input, il programma CGI per il trattamento degli stessi (stavolta scritto in linguaggio PERL) e il documento di risposta creato da tale programma.

[illegible]

---

CORSO DI HTML



## INPUT DEI DATI

Inserire il cognome	<input type="text" value="Fantozzi"/>
Inserire il nome	<input type="text" value="Ugo"/>
Inserire la professione	<input type="text" value="Impiegato"/>
Vuoi informazioni via e-mail ?	Si <input checked="" type="checkbox"/> No <input type="checkbox"/>
Indirizzo e-mail	<input type="text" value="fantozzi@aaa.it"/>
<input type="button" value="Invio"/>	<input type="button" value="RESET"/>

Programma CGI cgi-modulo.pl:

```
#!/usr/bin/perl
#
# cgi-modulo.pl
#
#=====
# Funzione messaggio
#=====
sub messaggio {
    print ("<P>Clicca <A HREF=$ENV{'HTTP_REFERER'}> qui</A> per tornare a
    correggere\n");
    return 0
};
#=====
# Main
#=====
$cognome = $ENV{'WWW_cognome'};
$nome = $ENV{'WWW_nome'};
$professione = $ENV{'WWW_professione'};
$risp = $ENV{'WWW_risp'};
$email = $ENV{'WWW_email'};
print ("Content-type: text/html\n");
print ("\n");
print ("<HTML>\n");
print ("<HEAD>\n");
print ("<TITLE>Test CGI</TITLE>\n");
print ("</HEAD>\n");
print ("<BODY>\n");
print ("<H1>Resoconto immissione dati</H1>\n");
if ($cognome eq "") {
    print ("<P>Cognome obbligatorio\n");
    &messaggio;
}
else {
    if ($nome eq "") {
        print ("<P>Nome obbligatorio\n");
        &messaggio;
    }
}
```





```
else {
    if (($risp eq "si") && ($email eq "")) {
        print ("<P>Se vuoi e-mail inserisci il tuo indirizzo\n");
        &messaggio;
    }
    else {
        print ("<P>Cognome: &#160;&#160;&#160;&#160;");
        print ("&#160;&#160;&#160;&#160; $cognome <P>\n");
        print ("Nome: &#160;&#160;&#160;&#160;&#160;&#160;&#160;");
        print ("&#160;&#160;&#160;&#160;&#160;&#160; $nome <P>\n");
        print ("Professione: &#160;&#160;&#160;&#160;");
        print ("&#160;$professione <P>\n");
        print ("Richiesta mail: $risp <P>\n");
        print ("Indirizzo: &#160;&#160;&#160;&#160;&#160;&#160;");
        print ("&#160;&#160;&#160;&#160;&#160;$email <P>\n");
    } }
print ("</BODY>\n");
print ("</HTML>\n");
```

Aspetto del documento di risposta creato da cgi-modulo.pl:

### Resoconto immissione dati

Cognome:	Fantozzi
Nome:	Ugo
Professione:	Impiegato
Richiesta mail:	si
Indirizzo:	fantozzi@aaa.it

### 4.3.6 Considerazioni finali

Come più volte fatto notare i programmi CGI vengono sempre eseguiti sul server WEB al quale ci si collega; ciò comporta alcuni limiti, in particolare:

- problemi di sicurezza: in quanto attraverso i programmi CGI alcuni utenti potrebbero accedere, volontariamente o meno, a dati confidenziali contenuti nel server;
- problemi di prestazioni: in quanto il server deve elaborare le richieste provenienti da più utenti anche contemporaneamente e quindi la velocità di risposta può diminuire anche drasticamente in caso di sovraccarico del sistema; inoltre le prestazioni dipendono fortemente dalla velocità della connessione tra macchina locale e server visto che tra esse c'è un continuo scambio di dati;
- problemi di accessibilità: in quanto, a causa dei problemi di sicurezza, molti gestori di siti WEB non consentono agli utenti di creare CGI personalizzati sul proprio server.

Ci sono però anche importanti vantaggi:

- compatibilità: in quanto i CGI funzionano con qualsiasi browser, essendo eseguiti sul server, e producendo risultati in formati universalmente riconosciuti come l'HTML;
- utilità: in quanto i CGI possono leggere e scrivere dati centralizzati sul server, raccogliere dati statistici sugli accessi, interrogare o aggiornare database.



A quest'ultimo aspetto, cioè l'interazione tra pagine WEB e database, che è molto importante, viene dedicato un approfondimento nel prossimo paragrafo.

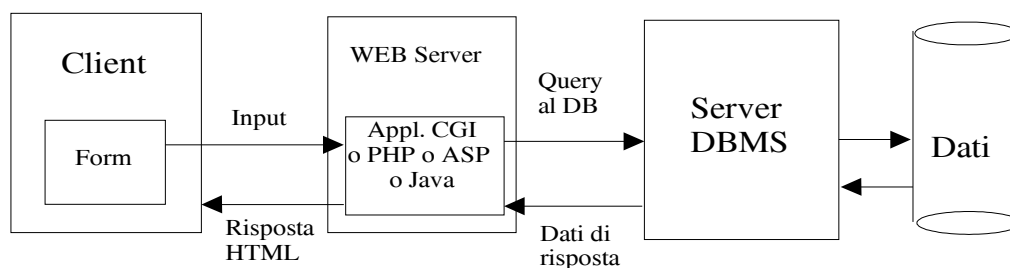
## 4.4 WEB e database, PHP

La possibilità di interfacciare un database relazionale con il WEB è senza dubbio assai interessante: da un lato infatti, abbiamo la potenza, l'affidabilità, la sicurezza dei moderni RDBMS (*Relational Data base Managment System*); dall'altro la facilità d'uso e la diffusione ormai universale degli strumenti di navigazione sia in Internet che nelle Intranet aziendali.

I campi di impiego di questa tecnologia sono numerosissimi: si pensi ad esempio alla possibilità di fornire ai cittadini l'accesso via Internet a dati di interesse pubblico o personale, oppure, all'interno di una azienda, alla gestione di database con interfacce basate sui normali, e molto amichevoli, programmi di navigazione.

Queste applicazioni si basano su una architettura a tre livelli: FRONT-END, MIDDLE-TIER, BACK-END.

Il primo livello è costituito dal form visualizzato con il browser sul client; il server WEB occupa il secondo livello mentre nel terzo livello risiede il database server.



Nei prossimi paragrafi esamineremo più da vicino i metodi e gli strumenti per utilizzare e gestire i dati di un database all'interno di un sito.

### 4.4.1 Strumenti necessari

Per la prima volta nell'ambito di questi appunti viene fatta una precisa scelta "di campo" relativamente agli strumenti da utilizzare, privilegiando il software OPEN SOURCE e segnatamente:

- Linux (distribuzione RedHat) come sistema operativo delle macchine che fungono da server WEB;
- Apache come server WEB;
- PostgreSQL come RDBMS;
- PHP come linguaggio di interfaccia tra WEB e database.

I motivi di questa scelta sono in parte "etici" (e qui non ci dilunghiamo in approfondimenti circa l'opportunità di usare software libero anziché proprietario in ambito educativo), in parte pratici in quanto con estrema facilità (e gratuitamente) si ha disposizione tutto il necessario per realizzare siti che utilizzano database. La scelta è altresì pienamente giustificata anche dal punto di vista tecnico essendo tali prodotti all'avanguardia nel loro settore e molto diffusi in ambito professionale. Apache, ad esempio è il server WEB nettamente più diffuso nel Mondo mentre PHP è un linguaggio di scripting in grado di rivaleggiare tranquillamente con il concorrente "proprietario" ASP.

In questa sede non vengono forniti dettagli sul reperimento e l'installazione dei pacchetti software citati; a tale proposito si rimanda ad appositi testi o alla documentazione presente anche in Internet.



Supponiamo quindi di avere già installato con successo tutto il software su un Personal computer su cui effettuare le prove e di avere il server Apache che risponde all'indirizzo `http://localhost`; inoltre diamo per scontato che il server sia completo dei moduli di supporto del linguaggio PHP.

Riguardo a PostgreSQL vengono introdotti i comandi necessari alla realizzazione di semplici esempi di gestione e interrogazione dati tramite il linguaggio PHP. Relativamente ai database relazionali in generale, data la vastità dell'argomento, si consiglia di ricorrere all'abbondante letteratura specialistica.

### 4.4.2 Database con PostgreSQL (cenni)

PostgreSQL è un Sistema di gestione basi dati relazionali molto avanzato e quasi del tutto rispondente allo standard SQL (Structured Query Language). E' stato sviluppato originariamente dal dipartimento di informatica dell'Università di Berkeley in California ed è di pubblico dominio e open source.

Una volta installato PostgreSQL con successo si può iniziare ad usarlo senza grosse difficoltà. La directory in cui vengono salvati gli archivi è per default `/var/lib/pgsql` e l'amministratore del database è l'utente "postgres" al quale è consigliabile assegnare una password (operazione che deve essere svolta dall'utente "root" di linux).

Prima di iniziare si deve attivare il programma di gestione del database con il comando (eseguito sempre da root):

```
# /etc/rc.d/init.d/postgresql start
```

Essendo la prima volta che viene eseguito il programma di amministrazione dei database, vengono creati i file fondamentali per la loro gestione.

In vista delle esecuzioni successive è opportuno inserire il servizio postgresql tra quelli attivati automaticamente al boot della macchina mediante il comando "setup".

Per testare l'installazione collegandosi come root si crea un utente "fulvio":

```
# adduser fulvio
# passwd fulvio
```

Poi ci si connette come postgres e si crea lo stesso utente per PostgreSQL:

```
$ createuser fulvio
```

Il sistema pone a questo punto delle domande relative al nuovo utente: si risponde in modo che non sia abilitato a creare database e non sia superuser e si accetta la creazione automatica di un database che egli possa gestire.

Finalmente è il momento di collegarsi come "fulvio" ed eseguire "psql" che è il programma di interfaccia, in modalità testuale, verso il database.

Si può creare una tabella per le prove di nome rubrica dove archiviare alcuni indirizzi:

```
fulvio=> create table rubrica(cognome varchar(20), nome varchar (20),
fulvio-> indirizzo varchar (40), telefono varchar (15), email varchar (30));
```

come si vede il comando può essere spezzato su più righe e non è concluso con il tasto "Invio" ma con il simbolo ";" la seconda riga è quindi una riga di continuazione e ciò è evidenziato anche dal prompt che si chiude con i simboli "->" invece che con "=>".

Per inserire un paio di record si usano i seguenti comandi:

```
fulvio=> insert into rubrica values('Rossi', 'Mario',
```



```
fulvio-> 'via Roma 34 Treviso', '04220101010', 'marioro@aaaa.it');
```

```
fulvio=> insert into rubrica values('Paperino', 'Paolino',  
fulvio-> 'via Roma 1 Paperopoli', '010011234567', 'paperino@abcd.com');
```

Verifichiamo i dati inseriti con il comando:

```
fulvio=> select * from rubrica;
```

che ci dovrebbe dare il seguente risultato:

cognome	nome	indirizzo	telefono	email
-----+-----+-----+-----+-----				
Rossi	Mario	via Roma 34 Treviso	04220101010	marioro@aaaa.it
Paperino	Paolino	via Roma 1 Paperopoli	010011234567	paperino@abcd.com
(2 rows)				

Le stesse operazioni possono essere fatte mediante il programma "PgAccess" (incluso in Postgres) che fornisce un'interfaccia grafica più amichevole.

Non approfondiamo ulteriormente l'uso di questi strumenti e "conserviamo" il piccolo archivio creato per utilizzarlo, nel proseguo, come database da interfacciare con pagine WEB.

### 4.4.3 Il linguaggio PHP (cenni)

Il PHP (*Personal Home Page*) nasce nel 1994 per mano di Rasmus Lerdorf ed è oggi utilizzato in molte decine di migliaia di siti in tutto il mondo; essendo un linguaggio di scripting non è dotato di "vita propria" ma viene utilizzato per estendere le possibilità dell'HTML. Viene eseguito dal lato server ed è embedded cioè incluso nei documenti HTML. Si presta molto bene per scrivere pagine WEB con funzioni di amministrazione di database. I database supportati sono PostgreSQL, MySQL, Oracle, Sybase, Informix. La versione più diffusa è la 3 e di questa ci occupiamo, anche se è ormai disponibile il nuovo PHP4.

I comandi PHP si inseriscono all'interno di un documento HTML in qualunque posizione e si distinguono dai normali tag di quest'ultimo racchiudendoli tra le stringhe "<?php" e ">". I documenti contenenti comandi PHP devono avere necessariamente l'estensione "php" o "php3" e non "html".

Come primo esempio definiamo nella nostra DOCUMENT ROOT (presumibilmente /home/httpd/html) un file "prova1.php" con l'unica linea:

```
<?php phpinfo() ?>
```

Se si accede al file con un browser si ottiene una risposta contenente numerose informazioni riguardanti il PHP, il proprio server WEB ed altro ancora.

Il linguaggio PHP può essere utilizzato anche indipendentemente dalla connessione a database come evidenziato dal seguente esempio in cui è presente un documento HTML per l'immissione di dati arricchito con alcuni comandi PHP; i dati di input vengono passati ad un altro file e trattati ancora con istruzioni PHP.

Il sorgente del primo documento è:

```
<HTML>  
<HEAD><TITLE>Esempio 1 con php</TITLE></HEAD>
```



```
<BODY>
<H3>Modulo gestito con PHP</H3>
<P><BR><P>
Sei collegato al server WEB:
<B><?php echo $SERVER_NAME; ?></B>
&nbsp;all'indirizzo:
<B><?php echo $SERVER_ADDR; ?> </B>
<P>Server powered by:
<B><?php echo $SERVER_SOFTWARE; ?></B>
<P>Sei collegato dall'indirizzo:
<B><?php echo $REMOTE_ADDR; ?></B>
<P><BR><P>
Inserisci i tuoi dati anagrafici
<P><FORM ACTION="/php1r.php" METHOD="POST">
  <P>Cognome: <INPUT TYPE="TEXT" NAME="cognome" SIZE="20" MAXLENGTH="20">
  <P>Nome: <INPUT TYPE="TEXT" NAME="nome" SIZE="20" MAXLENGTH="20">
  <P>Eta':
  <P><INPUT TYPE="RADIO" NAME="eta" VALUE="1-18">1-18
  <P><INPUT TYPE="RADIO" NAME="eta" VALUE="19-40">19-40
  <P><INPUT TYPE="RADIO" NAME="eta" VALUE="41-65">41-65
  <P><INPUT TYPE="RADIO" NAME="eta" VALUE="66-99">66-99
  <P>E-mail: <INPUT TYPE="TEXT" NAME="email" SIZE="30" MAXLENGTH="30">
  <P><INPUT TYPE="RESET" VALUE="Reset">
  <INPUT TYPE="SUBMIT">
</FORM></BODY>
</HTML>
```

Il suo aspetto visualizzato dal browser è il seguente:

## Modulo gestito con PHP

Sei collegato al server WEB: **ferronif.inf.besta** all'indirizzo: **127.0.0.1**

Server powered by: **Apache/1.3.12 (Unix) (Red Hat/Linux) PHP/3.0.15 mod\_perl/1.21**

Sei collegato dall'indirizzo: **127.0.0.1**

Inserisci i tuoi dati anagrafici

Cognome:

Nome:

Eta':

☐ 1-18

☒ 19-40

☐ 41-65

☐ 66-99

E-mail:

Reset

SUBMIT



Commentiamo brevemente i comandi di PHP inseriti in questo esempio.

Come già accennato il PHP è un linguaggio embedded ed infatti le istruzioni sono inserite all'interno del documento HTML racchiuse tra i tag speciali `<?php` e `?>`.

Ogni istruzione viene conclusa con `;`.

Si possono inserire commenti su un'unica linea facendoli precedere da `//` oppure multilinea racchiudendoli tra `/*` e `*/`.

Nella prima parte del documento vengono visualizzate informazioni sulla connessione grazie all'istruzione **echo** e all'uso di alcune variabili di ambiente (SERVER\_NAME, SERVER\_ADDR, ecc.).

Il resto del documento contiene la definizione del modulo per la richiesta di semplici dati anagrafici.

Si noti che i dati del modulo vengono inviati a "php1r.php" che risiede nella stessa directory del documento di partenza e non ad un eseguibile memorizzato in /cgi-bin e scritto in C, PERL o in altri linguaggi non embedded come invece avveniva nel caso di form gestiti con programmi CGI.

Il documento di risposta "php1r.php" non è altro che un'ulteriore pagina HTML contenente istruzioni PHP; il suo contenuto è:

```
<HTML>
<HEAD>
  <TITLE>Risposta esempio 1 con php</TITLE>
</HEAD>
<BODY>
  <H3>Risposta gestita con PHP</H3>
  <P><BR><P>
  Sei collegato al server WEB:
  <B><?php echo $SERVER_NAME; ?></B>
  &nbsp;all'indirizzo:
  <B><?php echo $SERVER_ADDR; ?> </B>
  <P>
  Server powered by:
  <B><?php echo $SERVER_SOFTWARE; ?></B>
  <P>
  Sei collegato dall'indirizzo:
  <B><?php echo $REMOTE_ADDR; ?></B>
  <P><BR><P>
  Hai inserito i seguenti dati:
  <P>Cognome: <?php echo $cognome; ?>
  <P>Nome: <?php echo $nome; ?>
  <P>Eta': <?php echo $eta; ?>
  <P>E-mail: <?php echo $email; ?>
  <P><BR><P><B>
  <?php

  $var1 = strstr($HTTP_USER_AGENT, "Mozilla");
  if (!strstr($var1, "compatible") and ($var1)) {
    echo "Stai usando Netscape Navigator"; }
  elseif (strstr($var1, "MSIE")) {
    echo "Stai usando Internet Explorer"; }
  else {
    echo "Stai usando un browser diverso da Explorer e Navigator"; }
?>
</B>
```



</BODY>  
</HTML>

E viene così visualizzato:

## Risposta gestita con PHP

Sei collegato al server WEB: **ferronif.inf.best** all'indirizzo: **127.0.0.1**

Server powered by: **Apache/1.3.12 (Unix) (Red Hat/Linux) PHP/3.0.15 mod\_perl/1.21**

Sei collegato dall'indirizzo: **127.0.0.1**

Hai inserito i seguenti dati:

Cognome: Rossi

Nome: Mario

Eta': 19-40

E-mail: rossima@aaaaa.it

**Stai usando un browser diverso da Explorer e Navigator**

La prima parte ricalca quella del primo documento; successivamente vengono visualizzati i dati inviati dal modulo.

Qui si nota un'altra differenza fondamentale rispetto ai programmi CGI: i dati inviati sono già a disposizione con i nomi definiti nel modulo (grazie all'attributo NAME) e non occorre alcuna operazione di decodifica né tantomeno l'uso della variabile QUERY\_STRING o la lettura dello standard input. Inoltre è del tutto ininfluenza che il metodo usato per l'invio sia "POST" oppure "GET".

Nell'ultima parte del sorgente c'è un blocco di codice PHP più complesso:

prima viene definita una variabile con nome **\$var1** (notare che i riferimenti a tutte le variabili in PHP avvengono facendo precedere il nome dal simbolo \$); il suo contenuto è il risultato della funzione PHP **strstr** definita sulla stringa \$HTTP\_USER\_AGENT (altra variabile di ambiente contenente il nome del browser usato) con argomento "Mozilla". Il risultato di tale funzione è la stringa ottenuta da quella di partenza escludendo la parte precedente la parola "Mozilla".

Successivamente viene utilizzata l'istruzione **if.....elseif....else....** sulla quale è opportuno soffermarsi più a lungo: le condizioni da verificare sono racchiuse tra parentesi tonde, il simbolo ! significa negazione, il blocco di istruzioni da eseguire nel caso di condizione verificata è racchiuso tra parentesi graffe.

La logica di questa istruzione è la seguente:

se **\$var1** è diversa dalla stringa vuota e non contiene la stringa "compatible" stiamo usando il browser "Mozilla" (o Netscape Navigator), altrimenti se **\$var1** contiene la stringa "MSIE" stiamo usando Internet Explorer, altrimenti stiamo usando un browser diverso da entrambi.



### 4.4.4 Funzioni PostgreSQL di PHP (cenni)

Il linguaggio PHP comprende molte funzioni da utilizzare per lavorare con PostgreSQL; l'elenco completo si può trovare nel manuale (<http://www.php.net/manual>). Per i nostri scopi attuali è sufficiente conoscere solo alcune di queste funzioni e precisamente:

- `pg_connect` - apre una connessione a PostgreSQL; richiede come parametri un nome-host, il nome del database, nome utente e password;
- `pg_exec` - esegue un'istruzione SQL;
- `pg_numrows` - restituisce il numero di record contenuti nel risultato di un'istruzione;
- `pg_result` - ritorna dei valori estratti da una tabella di interrogazione ottenuta con `pg_exec`;
- `pg_errormessage` - restituisce l'eventuale messaggio di errore dopo una operazione senza buon esito su una certa connessione;
- `pg_close` - chiude la connessione corrente.

Nell'esempio seguente viene realizzato un modulo per inserire i dati nella base dati "rubrica" definita in precedenza con PostgreSQL. Con lo stesso modulo è anche possibile effettuare un'interrogazione all'archivio specificando alcuni parametri. I dati da inserire o da usare per la ricerca vengono passati ad un file PHP che effettua l'operazione richiesta e crea una opportuna pagina di risposta.

Il sorgente del modulo è definito come segue (il file può avere estensione ".html" in quanto non contiene istruzioni PHP):

```
<HTML>
<HEAD>
<TITLE>Gestione Base Dati Rubrica</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<CENTER><H3>Gestione Rubrica</H3></CENTER>
<FORM METHOD="GET" ACTION="/phpsql.php">
<P><B>Cognome: </B>
<INPUT TYPE="text" NAME="cognome" MAXLENGTH=25 size=20> **<P>
<P><B>Nome &#160;&#160;&#160;&#160;&#160;&#160;: </B>
<INPUT TYPE="text" NAME="nome" MAXLENGTH=25 SIZE=20> **<P>
<P><B>Indirizzo &#160;: </B>
<INPUT TYPE="text" NAME="indirizzo" MAXLENGTH=40 SIZE=40>
<P><BR><P>
<P><B>Telefono &#160;: </B>
<INPUT TYPE="text" NAME="telefono" MAXLENGTH=15 SIZE=15> **
<P><BR><P>
<P><B>E-mail &#160;&#160;&#160;&#160;&#160;: </B>
<INPUT TYPE="text" NAME="email" MAXLENGTH=30 SIZE=30>
<P><BR><P>
** = campi validi per l'interrogazione
(lasciarli <B>tutti vuoti </B>per ottenere la <B>lista completa </B>dell'archivio)
<P><BR><P>
<TABLE><TR><TD><INPUT TYPE="reset" VALUE="Azzera"></TD><TD
WIDTH=30></TD>
<TD><INPUT TYPE="submit" VALUE="Inserisci" NAME="sub"></TD>
<TD><INPUT TYPE="submit" VALUE="Interroga" NAME="sub"></TD></TR></TABLE>
</FORM>
<P>
</BODY>
</HTML>
```





Viene visualizzato in questo modo:

## Gestione Rubrica

**Cognome:**  \*\*

**Nome :**  \*\*

**Indirizzo :**

**Telefono :**  \*\*

**E-mail :**

\*\* = campi validi per l'interrogazione (lasciarli **tutti vuoti** per ottenere la **lista completa** dell'archivio)

N.B. In caso di inserimento il cognome è obbligatorio.

Il file "phpsql.php" che svolge le operazioni sul database è il seguente:

```
<HTML>
<HEAD>
<TITLE>Interrogazione / inserimento Rubrica</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<CENTER>

<?php
// imposta le variabili per la connessione al database e la apre

$PG_DATABASE="fulvio";
$PG_HOST="localhost";
$PG_PORT=5432;
$PG_USER="fulvio";
$PG_PSW="";

$conn=pg_connect("dbname=$PG_DATABASE host=$PG_HOST port=$PG_PORT
user=$PG_USER password=$PG_PSW");
```



```
if (! $conn) {
    echo "<P><B>Connessione al database non riuscita</B>";
    exit(); }

// verifica se richiesto un inserimento o una interrogazione

if ($sub == "Interroga") {
    echo "<P><H3>Risultato dell'interrogazione</H3></P></CENTER>";
    // interrogazione: prepara la variabile alt in base ai parametri richiesti

    $salt=0;
    if ($cognome != "") $salt=100;
    if ($nome != "") $salt += 10;
    if ($telefono != "") $salt += 1;

    /* L'istruzione switch permette di eseguire l'interrogazione appropriata in
    base all'impostazione dei parametri */

    switch ($salt)
    {
    case 111:
        $richiesta=pg_exec($conn,"SELECT * FROM rubrica WHERE cognome='$cognome' AND
        nome='$nome' AND telefono='$telefono'");
        $righe=pg_numrows($richiesta);
        break;

    case 110:
        $richiesta=pg_exec($conn,"SELECT * FROM rubrica WHERE cognome='$cognome' AND
        nome='$nome'");
        $righe=pg_numrows($richiesta);
        break;

    case 101:
        $richiesta=pg_exec($conn,"SELECT * FROM rubrica WHERE cognome='$cognome' AND
        telefono='$telefono'");
        $righe=pg_numrows($richiesta);
        break;

    case 100:
        $richiesta=pg_exec($conn,"SELECT * FROM rubrica WHERE cognome='$cognome'");
        $righe=pg_numrows($richiesta);
        break;

    case 11:
        $richiesta=pg_exec($conn,"SELECT * FROM rubrica WHERE nome='$nome'
        AND telefono='$telefono'");
        $righe=pg_numrows($richiesta);
        break;

    case 10:
        $richiesta=pg_exec($conn,"SELECT * FROM rubrica WHERE nome='$nome'");
        $righe=pg_numrows($richiesta);
        break;

    case 1:
```



```
$richiesta=pg_exec($conn,"SELECT * FROM rubrica WHERE telefono='$telefono'");
$righe=pg_numrows($richiesta);
break;

default:
    $richiesta=pg_exec($conn,"SELECT * FROM rubrica;");
    $righe=pg_numrows($richiesta);
} // fine switch

/* se l'interrogazione ha restituito un risultato, da esso vengono estratti i
dati e vengono formattati nel documento di risposta */

if ($righe > 0)
{
    print "<UL>";
    $cont=0;
    while ($cont < $righe)
    { $v1 = pg_result($richiesta,$cont,"cognome");
      $v2 = pg_result($richiesta,$cont,"nome");
      $v3 = pg_result($richiesta,$cont,"indirizzo");
      $v4 = pg_result($richiesta,$cont,"telefono");
      $v5 = pg_result($richiesta,$cont,"email");
      print "<P>Cognome   : ";
      print $v1;
      print "<BR>Nome     : ";
      print $v2;
      print "<BR>Indirizzo : ";
      print $v3;
      print "<BR>Telefono  : ";
      print $v4;
      print "<BR>Email    : ";
      print $v5;
      print "<P>";
      $cont++;
    } // fine while
    print "</UL>";
}
else
    print "<P><CENTER><B>Nella base dati non presente alcuna informazione<BR>
con le caratteristiche specificate<B><CENTER><P>";
} // fine if ($righe)

else
// Inserimento: il cognome deve essere indicato
{
    if (! $cognome)
        echo "<P><B> Il cognome &grave; obbligatorio";
    else
    {
        $inser = "insert into rubrica values ('$cognome', '$nome', '$indirizzo',
                                                '$telefono', '$email')";

        $ris = pg_exec($conn, $inser );
        if ( ! $ris )
            echo "Errore : " + pg_errormessage( $conn );
        else

```



```
echo "Inserimento di $cognome $nome effettuato";
} // fine if (! $cognome)

} // fine if ($sub.....)

pg_close($conn);
print "<P> </P>";
print "<CENTER><HR WIDTH=30%><P>
<A HREF=\"./moduloxphp.html\"><B>Nuova operazione</B></A> </CENTER>";
?>

</BODY>
</HTML>
```

Nel caso venga richiesta una interrogazione fornendo come unico parametro il cognome "Rossi" si ottiene la risposta riportata sotto (naturalmente si suppone che prima siano stati inseriti anche i dati di Rossi Andrea oltre a quelli di Rossi Mario già presenti):

### Risultato dell'interrogazione

Cognome : Rossi  
Nome : Mario  
Indirizzo : via Roma 34 Treviso  
Telefono : 04220101010  
Email : mario@aaaa.it

Cognome : Rossi  
Nome : Andrea  
Indirizzo : -  
Telefono : 0422888888  
Email : andrea@aaaa.it

---

**Nuova operazione**

La stringa "Nuova operazione" è un link per tornare al modulo di partenza.

Sebbene il sorgente "phpsql.php" contenga dei commenti che illustrano alcuni dei punti più significativi dell'elaborazione, è senz'altro opportuno spiegarne ulteriormente la logica di funzionamento soffermandosi specialmente sulle funzioni di collegamento al database.

Nella prima parte vengono definite le variabili contenenti il nome del computer, del database, dell'utente, la password, che vengono usate nella funzione di connessione al database **pg\_connect**. La variabile **\$conn** contiene il risultato del tentativo di connessione e viene quindi subito testata per verificare che l'operazione sia andata a buon fine.

Con il test sulla variabile **\$sub** si stabilisce se l'utente ha richiesto una interrogazione o un inserimento.

Nel primo caso viene preparata la variabile **\$var** in base ai parametri di ricerca impostati; poi tale variabile viene testata con l'istruzione **switch** tramite la quale si esaminano le varie combinazioni possibili di parametri di ricerca che l'utente può avere impostato.

In ognuno dei casi esaminati (istruzioni **case....**) viene fatta l'interrogazione alla base dati con la funzione **pg\_exec**: essa richiede il nome della connessione (**\$conn**) e l'operazione da svolgere espressa secondo la



sintassi dell'SQL, in una forma molto vicina alla lingua inglese e quindi facilmente comprensibile. La variabile **\$richiesta** contiene il risultato dell'interrogazione e **\$righe** viene caricata con il numero di elementi trovati grazie alla funzione **pg\_numrows**; l'istruzione **break** serve a concludere quel particolare blocco **case**. Il blocco **default** viene eseguito se il valore di **\$var** non coincide con nessuno dei "case" previsti. Successivamente con il test su **\$righe** si stabilisce se si sono estratti dei record dall'archivio (altrimenti si emette un opportuno messaggio); in caso affermativo si esegue un ciclo con l'istruzione **while** tante volte quante sono le righe estratte. Il contatore del ciclo è **\$scont** e viene incrementato di una unità con l'istruzione **\$scont++**. Per ogni iterazione si estraggono i dati con la funzione **pg\_result** che richiede il nome della variabile che contiene il risultato dell'interrogazione (**\$richiesta**), il numero di riga (**\$scont**) e il nome del campo che si vuole estrarre. Tali dati vengono poi emessi sulla pagina di risposta con le istruzioni **print**.

Nel caso invece l'utente abbia richiesto un inserimento, si controlla che sia stato indicato almeno il cognome e, in caso affermativo, si inseriscono in archivio i dati provenienti dal modulo grazie alla funzione **pg\_exec** per la quale viene impostata l'operazione richiesta nella variabile **\$inser**. La bontà dell'inserimento è verificata con il test su **\$ris**; in caso di problemi si emette l'errore verificatosi grazie all'uso della funzione **pg\_errormessage**.

A questo punto si conclude anche il ramo **else** della istruzione **if** iniziale sulla variabile **\$sub**.

Non resta quindi altro che chiudere la connessione al database con la funzione **pg\_close** ed emettere sul documento di risposta la stringa che funge da link per tornare al modulo di partenza.

Si noti infine l'uso del carattere di escape "\" nell'ultima **print** per fare in modo che le virgolette usate per il valore di HREF non vengano interpretate come virgolette di inizio e fine stringa da parte di PHP.

## 4.5 Script attivi, DHTML, applet, servlet

### 4.5.1 Java e JavaScript

Gli *script attivi* e le *applet* sono programmi o porzioni di codice eseguiti dal lato client e sono molto usati per rendere più dinamiche e interattive le pagine WEB.

Le applet sono chiamate anche *applet Java* perché scritte quasi sempre in linguaggio Java e vengono eseguite in una porzione di un documento WEB senza possibilità di interferire con ciò che avviene al di fuori della finestra di esecuzione.

Gli script attivi invece possono intervenire sulle pagine WEB modificandole e anche creandone di nuove.

Esistono fondamentalmente due linguaggi per la scrittura di script attivi: JavaScript creato dalla Netscape (JScript nella versione della Microsoft) e VBScript (*Visual Basic scripting Edition*). Quest'ultimo è un linguaggio di script sviluppato da Microsoft per essere simile a Visual Basic e si rivolge esclusivamente agli utenti che usano piattaforme Windows essendo supportato solo da Internet Explorer. JavaScript invece è supportato da tutti i browser più importanti e quindi continuiamo ad occuparci solo di quest'ultimo.

Prima di proseguire è però opportuno chiarire le differenze che ci sono, e sono notevoli, tra Java e JavaScript malgrado la somiglianza fra i nomi (in effetti il nome originale di JavaScript era LiveScript e fu cambiato per motivi di "marketing"):

- Java è un linguaggio compilato mentre JavaScript è interpretato;
- le applicazioni Java possono essere anche indipendenti ed essere eseguite fuori dalle pagine WEB, il codice JavaScript invece è sempre legato ad un documento HTML e non può essere eseguito separatamente (da questo punto di vista tra Java e JavaScript c'è la stessa differenza esistente tra PERL e PHP);
- le applet Java sono confinate in una ben definita regione della pagina, JavaScript invece può controllare un'intero documento e rispondere alla pressione di pulsanti, click del mouse o altri eventi simili;



- le applet Java sono programmi separati dal codice HTML, vengono scaricati dalla rete ed eseguiti da un browser compatibile con Java; il codice JavaScript è invece generalmente embedded cioè inserito all'interno del sorgente HTML ed è eseguito da un browser compatibile con JavaScript.

Ciò che accomuna le applet Java e i JavaScript è invece il fatto che, per motivi di sicurezza, non possono leggere o scrivere file. Questa attenzione alla sicurezza differenzia fondamentalmente le applet Java dalla tecnologia concorrente degli *ActiveX* di Microsoft.

ActiveX non è un linguaggio di programmazione ma un formato per dati ricavato da due tecnologie Microsoft esistenti: OLE (*Object Linking and Embedding*) che permette ai programmi di comunicare con altre applicazioni e COM (*Component Object Model*) che è un formato per la definizione di interazioni tra oggetti, poi evolutasi nella tecnologia DCOM (*Distributed COM*) per la definizione di interazioni remote tra oggetti.

Gli Activex possono essere scritti in molti linguaggi ma sono molto difficili da includere nelle pagine WEB senza usare altri strumenti Microsoft, inoltre sono supportati solo da Internet Explorer e, cosa più grave, possono interagire con la macchina client salvando o modificando file, lanciando l'esecuzione di altri programmi, aprendo documenti ed altro ancora.

E' ovvio quindi che questi strumenti possono rappresentare un pericolo, se non usati con i dovuti accorgimenti, riguardo ad esempio alla diffusione attraverso Internet di virus informatici o alla violazione della riservatezza dei dati contenuti nel computer di un utente.

## 4.5.2 Script attivi (cenni)

JavaScript è un linguaggio di script embedded sviluppato da Netscape (originariamente con il nome di LiveScript).

L'elemento da utilizzare è `<SCRIPT>` che può essere posizionato sia nella sezione HEAD che nella sezione BODY di un documento HTML e prevede il tag di chiusura `</SCRIPT>`. Le istruzioni JavaScript si scrivono tra i due tag ma è opportuno "mascherarle" ulteriormente come commenti HTML per evitare che vecchi browser, per errore, le mostrino insieme al contenuto della pagina. Quindi il codice JavaScript inizia di solito con `<!-- Begin script` e si conclude con `"/>`.

Si noti che i commenti di JavaScript si inseriscono con i caratteri `"/>` oppure `"/>` e `"/>` nel caso di commenti multilinea.

L'elemento `<SCRIPT>` prevede come attributi `LANGUAGE` e `TYPE`; usando JavaScript i valori da assegnare sono rispettivamente: `"JavaScript"` e `"text/javascript"`.

Un altro attributo è `SCR` con il quale si può fare riferimento ad uno script esterno che viene caricato dal browser ed eseguito. Gli script esterni (da non confondere assolutamente con gli script CGI di cui abbiamo parlato in precedenza) risiedono su file separati che hanno di solito estensione `".js"`, contengono esclusivamente comandi JavaScript e possono essere utili in quanto permettono di modificare una volta sola il codice usato in molti documenti HTML diversi. Si tenga però conto dell'aggravio di tempo necessario a ricevere dal server anche il file contenente lo script oltre alla pagina WEB che lo richiama e del fatto che alcuni server WEB possono non essere configurati per un corretto invio degli script esterni.

Vediamo adesso un semplice esempio in cui si usa una funzione scritta in JavaScript per far immettere all'utente il proprio nome e poi presentarlo in una pagina di risposta.

```
<HTML>
<HEAD><TITLE>Esempio con JavaScript</TITLE>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
  <!--Begin script
  function InsNome() {
    nomeut = prompt("Inserire il proprio nome:", " ");
    if (nomeut == "" || nomeut == null) {
      nomeut = "Utente sconosciuto";
    }
    return (nomeut);
  }
</SCRIPT>
```



```
}  
// -->  
</SCRIPT>  
</HEAD>  
<BODY>  
<DIV ALIGN="CENTER"><B>Esempio con JavaScript</B></DIV>  
<P><BR><P>  
Il nome inserito &grave;;:  
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">  
  <!--Begin script  
    document.write (InsNome());  
  // -->  
</SCRIPT>  
</BODY>  
</HTML>
```

Nella prima parte viene definita una funzione chiamata **InsNome()**; tale funzione non necessita di alcun parametro e serve a far inserire all'utente il proprio nome con l'istruzione **prompt**. Il nome inserito viene poi testato con l'istruzione **if** (i simboli **||** all'interno del test corrispondono alla **or**) e valorizzato automaticamente se è vuoto. L'ultima istruzione della funzione è **return** che serve a restituire il risultato della funzione stessa, in questo caso il nome inserito dall'utente.

Nella seconda parte del documento c'è un altro blocco di codice JavaScript: in esso viene richiamata la funzione **InsNome()** all'interno dell'istruzione **document.write** con la quale il nome inserito viene scritto all'interno della pagina HTML.

La visualizzazione di questo documento avviene secondo la seguente logica: viene iniziata la visualizzazione della sezione **BODY**, poi viene eseguito il codice JavaScript all'interno di **BODY** e quindi l'istruzione di scrittura. All'interno di quest'ultima c'è però il richiamo alla funzione **InsNome()** e quindi, prima che la scrittura sia ultimata, appare una piccola maschera con la richiesta del nome (effetto dell'istruzione **prompt**). Ad inserimento effettuato la maschera di input scompare e il nome inserito viene visualizzato insieme al resto del contenuto della sezione **BODY**.

### 4.5.2.1 *Attributi di evento*

Nell'HTML 4.0 sono disponibili una serie di attributi di evento che si possono abbinare ad immagini, collegamenti e ad altri elementi; ad ogni evento si può poi associare del codice JavaScript da eseguire nel momento in cui l'evento in questione si verifica.

Vediamo quali sono gli attributi di evento iniziando da quelli più largamente utilizzabili:

**ONCLICK**: si verifica quando si clicca con il mouse su un elemento;  
**ONDBCLICK**: quando si fa doppio clic con il mouse su un elemento;  
**ONKEYPRESS**: quando un tasto viene premuto e rilasciato su un elemento;  
**ONKEYDOWN**: quando un tasto viene premuto su un elemento;  
**ONKEYUP**: quando un tasto viene rilasciato su un elemento;  
**ONMOUSEDOWN**: si verifica quando si clicca con il mouse su un elemento (senza rilasciare);  
**ONMOUSEUP**: si verifica quando si rilascia il pulsante del mouse su un elemento;  
**ONMOUSEMOVE**: quando il puntatore del mouse si muove sopra ad un elemento;  
**ONMOUSEOVER**: quando il puntatore del mouse passa su un elemento provenendo da "fuori" di esso;  
**ONMOUSEOUT**: quando il puntatore del mouse "lascia" un elemento.

I seguenti attributi si possono usare solo all'interno dei **FORM**:

**ONSUBMIT**: si verifica quando un modulo viene inviato (si usa solo nel tag **<FORM>**);  
**ONRESET**: quando il contenuto di un modulo viene ripristinato (si usa solo nel tag **<FORM>**);  
**ONSELECT**: quando si seleziona del testo (si usa solo con i tag **<INPUT>** e **<TEXTAREA>**);



ONCHANGE: quando cambia il valore di un controllo (si usa solo con i tag <INPUT>, <TEXTAREA> e <SELECT>).

I seguenti attributi si possono usare con i FORM e con i collegamenti:

ONFOCUS: si verifica quando un elemento viene selezionato (si dice che "riceve il fuoco") con il mouse o la tastiera;

ONBLUR: quando un elemento perde il fuoco.

Infine i seguenti attributi si possono usare solo nei tag <BODY> e <FRAMESET>:

ONLOAD: si verifica quando il navigatore termina il caricamento del documento;

ONUNLOAD: quando il navigatore accede ad un altro documento.

Per usare correttamente gli attributi di evento si deve definire il linguaggio di script di default del documento: a questo scopo si usa il seguente elemento <META>:

```
<META HTTP-EQUIV="Content-Script-Type" CONTENT="text/javascript">
```

Nel seguente esempio abbiamo l'uso di JavaScript per controllare l'input in due campi di un form:

```
<HTML>
<HEAD><TITLE>Esempio con JavaScript</TITLE>
<META HTTP-EQUIV="Content-Script-Type" CONTENT="text/javascript">
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
  <!-- Begin script
  function Contr(campo) {
    if (campo.length > 0) {
      return (true);
    }
    else {
      alert("Campo " + campo.name + " obbligatorio!");
      return (false)
    }
  }
  // -->
</SCRIPT>
</HEAD>
<BODY>
<DIV ALIGN="CENTER"><B>Input gestito con JavaScript</B></DIV>
<P><BR><P>
<FORM ACTION="/cgi-bin/uncgi/cgi-var.sh" METHOD="GET">
  Cognome: <INPUT NAME="cognome" TYPE="TEXT" ONBLUR="Contr(cognome)"><P>
  Nome: <INPUT NAME="nome" TYPE="TEXT" ONBLUR="Contr(nome)"><P>
  <INPUT TYPE="SUBMIT" VALUE="INVIO">
</FORM>
</BODY>
</HTML>
```

Infine ecco un esempio in cui viene presentata una immagine cliccando sulla quale si apre un'altra pagina HTML:

```
<HTML>
<HEAD><TITLE>Finestra da aprire in JavaScript</TITLE>
<META HTTP-EQUIV="Content-script-type" CONTENT="text/javascript">
</HEAD>
```





```
<BODY>
<DIV ALIGN="CENTER"><B>Finestra da aprire con JavaScript</B></DIV>
<P><BR><P>
<A HREF="#" onclick='window.open("esejs3b.html",
    "Prova Javascript","scrollbars=no,resizable=no")'>
    <IMG SRC="gaia.jpg" WIDTH="80" HEIGHT="80"></A>
</BODY>
</HTML>
```

### 4.5.2.2 Disabilitazione dei JavaScript

L'uso di JavaScript offre senz'altro grandi possibilità (che gli esempi appena illustrati mostrano in misura molto modesta) ma non bisogna sottovalutare alcuni aspetti negativi:

- molti browser non supportano gli enunciati JavaScript e anche quelli che li supportano si comportano spesso in modo diverso di fronte ad una stessa istruzione;
- il linguaggio ha dei limiti dovuti ad importanti ragioni di sicurezza;
- JavaScript viene spesso usato per creare effetti che alla lunga possono risultare noiosi e fastidiosi; alcuni utenti preferiscono addirittura disabilitare, intervenendo su opportune opzioni dei browser, l'esecuzione degli script.

Per questi motivi è lecito aspettarsi che un documento contenete istruzioni JavaScript possa venire visualizzato da navigatori non in grado di eseguirle correttamente; è quindi opportuno prevedere subito dopo la fine di uno script, l'uso dei tag `<NOSCRIPT>` e `</NOSCRIPT>` all'interno dei quali inserire le istruzioni HTML alternative allo script stesso.

### 4.5.3 DHTML (cenni)

Il DHTML (*Dynamic HTML*) non è uno specifico linguaggio e neanche una evoluzione dell'HTML, è invece un insieme di tecniche utilizzabili in modo coordinato per costruire pagine WEB dinamiche "client-side".

Le tecnologie coinvolte nell'uso del DHTML sono:

- un linguaggio di scripting;
- i fogli di stile CSS;
- un modello ad oggetti del documento o DOM (*Document Object Model*) che permette di integrare le altre due tecnologie definendo un'interfaccia di programmazione indipendente da linguaggio e piattaforma per accedere a contenuto, stile e struttura di un documento DHTML; grazie al DOM è possibile gestire e controllare ogni singolo oggetto del documento con una opportuna sintassi derivata dalla programmazione ad oggetti.

Il consorzio W3C ha definito degli standard ufficiali per ognuna delle tre componenti:

- per il linguaggio di scripting lo standard è ECMAScript nato dal JavaScript 1.1 ed al quale comunque non si discostano di molto i successivi JavaScript 1.2 e 1.3 (da notare che invece il meno usato Jscript di Microsoft aderisce invece allo standard);
- per il CSS la versione ufficiale è dal 1998 la CSS-2;
- per il DOM, nel 1998 è stato pubblicato il "DOM Level 1 Recommendation".

Attualmente i DOM di Netscape Navigatore e Internet Explorer sono molto diversi tra loro e questo provoca notevoli problemi a livello di fruibilità universale dei documenti DHTML; talvolta gli sviluppatori sono addirittura costretti a creare versioni diverse della stessa pagina da visualizzare ognuna con il navigatore più appropriato.

Questa situazione dovrebbe comunque cessare visto che il DOM di Microsoft è in procinto di essere adottato dal W3C come DOM standard.



Maffiori dettagli sugli oggetti contenuti nei DOM e, più in generale, sull'uso delle tecniche del DHTML si possono reperire sui testi relativi all'argomento cui si fa riferimento al termine del capitolo.

## 4.5.4 Applet Java (cenni)

### 4.5.4.1 Caratteristiche del linguaggio Java

Java è un linguaggio di programmazione creato nel 1995 da Sun Microsystem (forse la più agguerrita concorrente di Microsoft) e deve il suo nome a una varietà di caffè tropicale. Inizialmente ideato per essere incorporato nei microchip che governano gli elettrodomestici si è poi affermato come linguaggio di programmazione soprattutto nell'ambito del WEB.

Uno dei principali motivi del successo di Java in ambito Internet è la sua sicurezza intrinseca: con esso si viene a creare un "firewall", cioè una barriera protettiva tra l'applicazione di rete e il computer dell'utente. Infatti quando si scaricano con un browser compatibile le applicazioni Java non ci sono rischi di infezione virale o di comportamenti illeciti da parte delle stesse applicazioni in quanto i programmi Java vengono confinati in un apposito ambiente di esecuzione senza alcuna possibilità di accedere ad altre parti del computer.

Altro aspetto fondamentale è quello della portabilità: un programma scritto in Java può essere eseguito indifferentemente su ogni sistema operativo e su ogni piattaforma senza subire modifiche.

Ciò che consente a Java di essere così sicuro e portabile è il fatto che l'output della compilazione di un sorgente di tale linguaggio non è codice eseguibile bensì "bytecode".

Un bytecode è un insieme ottimizzato di istruzioni che vengono eseguite da una "macchina virtuale" detta JVM (*Java Virtual Machine*) che viene emulata dall'ambiente di esecuzione di Java, attingendo alle risorse della macchina reale.

L'eseguibile non ha alcuna possibilità di "sconfinare" al di fuori della macchina virtuale; questo risolve i problemi di sicurezza. Inoltre è sufficiente realizzare su piattaforme diverse diversi sistemi run-time di Java, uno per ogni piattaforma, affinché su tutte possa girare qualsiasi programma Java; così anche la portabilità è assicurata.

L'ambiente di esecuzione (o run-time) di Java non è altro che un interprete del bytecode; il fatto che i programmi Java vengano interpretati e non compilati e poi eseguiti, comporta senza dubbio problemi a livello di prestazioni. Tali problemi però vengono largamente compensati, almeno a livello di applicazioni per il WEB, dai vantaggi riguardanti la sicurezza e la portabilità.

Altri motivi per cui Java è molto apprezzato dai programmatori sono:

- è semplice (almeno per chi già conosce il linguaggio C++);
- è orientato agli oggetti con un approccio molto "pulito" e utilizzabile (il tema della programmazione ad oggetti è di una tale vastità e complessità che scoraggia qualsiasi tentativo di semplificazione e sintesi; gli interessati a questo o anche ad approfondire lo studio del linguaggio Java possono consultare i testi dedicati a questi argomenti);
- è multithreaded vale a dire che consente di scrivere programmi che fanno più cose contemporaneamente in quanto il run-time di Java comprende un'elegante e sofisticata soluzione per la sincronizzazione multiprocesso;
- è solido e affidabile sia perché fortemente "tipizzato" (le variabili devono essere dichiarate prima di essere utilizzate e non sono possibili conversioni implicite di tipo), sia perché privo di puntatori (il cui uso da parte di programmatori non esperti è spesso fonte di errori), sia perché dotato di "automatic garbage collection" cioè di un sistema automatico che analizza la memoria e libera quella inutilizzata (in altri linguaggi come il C++ questa operazione è invece a carico del programmatore e può determinare errori);
- è distribuito in quanto è stato progettato appositamente per lo sviluppo di applicazioni distribuite; quindi un'applicazione Java può essere costituita da più moduli, residenti su diversi computer, in grado di operare congiuntamente attraverso una rete telematica.



### 4.5.4.2 Creazione ed esecuzione delle applet Java

Per creare una applet Java si devono compiere le seguenti operazioni:

prima di tutto si deve scrivere il sorgente Java salvandolo in un file con estensione ".java" e con nome uguale a quello della classe definita nel sorgente stesso (supponiamo sia "EseJava");  
poi si deve compilare il sorgente utilizzando un compilatore Java (ad esempio JDK - *Java Development Kit* disponibile gratuitamente presso <http://java.sun.com>; il comando è

```
javac EseJava.java
```

si ottiene un file con stesso nome ed estensione ".class"; nel nostro esempio "EseJava.class";

l'eseguibile ottenuto può anche essere eseguito indipendentemente da qualsiasi pagina WEB, tramite l'interprete java con il comando:

```
java EseJava.class
```

Per inserire invece l'applicazione Java all'interno di una pagina WEB si usa l'elemento di TESTO <APPLET> che prevede il tag di chiusura </APPLET>. Qualsiasi testo inserito tra questi due marcatori viene visualizzato nel caso il browser non sia compatibile con Java.

Vediamo gli attributi di questo elemento (i primi tre sono obbligatori):

CODE, per indicare il nome del file bytecode da eseguire;  
WIDTH e HEIGHT, per indicare le dimensioni in pixel dell'area rettangolare nella quale eseguire l'applet;  
HSPACE, VSPACE, ALIGN, usati come per le immagini;  
NAME, per assegnare un nome all'applet (può servire in caso più applet debbano comunicare tra loro se si usano tecniche Java avanzate come "LiveConnect" di Netscape);  
CODEBASE, indica la directory che contiene tutti i file a cui la applet fa eventualmente riferimento per il suo funzionamento.

Un altro elemento che si può usare con le applet è <PARAM> che serve ad indicare un parametro da passare all'applicazione Java. Si possono passare più parametri usando più volte questo tag e l'ordine con cui avviene il passaggio non è influente. Naturalmente si possono passare parametri solo ad una applet che sia stata scritta per accettarli e utilizzarli in qualche maniera. Di solito le applet già pronte sono documentate ed è quindi possibile sapere se o previsti parametri, con quali scopi e con quali nomi.

L'elemento <PARAM> si inserisce tra <APPLET> e </APPLET> e prevede due attributi:

NAME, per indicare il nome del parametro;  
VALUE, per indicare il valore del parametro.

Quando un documento contenente l'applet viene richiesto, quest'ultima viene inviata dal server insieme a tutti gli altri file multimediali: se il browser è in grado di interpretare il linguaggio, il programma viene eseguito. In questo modo le pagine WEB possono animarsi, integrare suoni in tempo reale, visualizzare video ed animazioni, presentare grafici dinamici ed altro ancora.

```
<HTML>
<HEAD><TITLE>Esempio con Java</TITLE>
</HEAD>
<BODY>
<DIV ALIGN="CENTER"><B>Esempio con applet Java</B></DIV>
<P><BR><P>
Qui sotto ci dovrebbe essere l'applet <P>
<APPLET CODE="EseJava.class" WIDTH="100" HEIGHT="100">
```



```
Java non va  
</APPLET>  
</BODY>  
</HTML>
```

Concludiamo con una osservazione circa il fatto che con HTML 4.0 è sconsigliato l'uso dell'elemento `<APPLET>` che dovrebbe essere sostituito da `<OBJECT>` di cui abbiamo parlato nel capitolo 3.

L'esempio precedente dovrebbe quindi essere inserito nella pagina WEB in questo modo:

```
<OBJECT CODETYPE="application/octet-stream" CLASSID="java:EseJava.class"  
        WIDTH="200" HEIGHT="250">  
Il Browser non supporta Java  
</OBJECT>
```

Non tutti i browser, specie le vecchie versioni, accettano questa sintassi per l'inserimento delle applet. Per il futuro è comunque da preferire l'uso dell'elemento `<OBJECT>` in quanto permette di inserire applicazioni scritte in linguaggi di programmazione diversi da Java.

### 4.5.5 Servlet Java e Java Server Pages

Le *servlet Java*, come suggerisce il nome, hanno delle analogie con le applet essendo applicazioni scritte in Java, ma vengono eseguite sul server anziché sul client. Esse vengono quindi attivate dalle pagine WEB esattamente come i programmi CGI con i vantaggi relativi (indipendenza dalla piattaforma client, possibilità di accedere a database ecc.) ma si differenziano (in meglio) anche da questi ultimi per il motivo seguente: in presenza di "normali" CGI il server HTTP esegue un processo ex-novo ogni volta che riceve una nuova richiesta di attivazione di quel programma; le servlet invece vengono eseguite e conservate in un determinato spazio di memoria in modo da potere essere richiamate in modo rapidissimo al sopraggiungere di successive richieste di esecuzione.

Per potere utilizzare le servlet Java occorre installare una estensione di Apache di nome *JServ* (reperibile presso <http://java.apache.org>). Al termine dell'installazione non dovrebbero essere necessarie altre operazioni o configurazioni particolari; si ricordi solo che le applicazioni servlet eseguibili devono risiedere per default nella directory `/home/http/servlets` anziché in `/home/http/cgi-bin`.

Il principale difetto delle servlet java è che esse, come le applicazioni CGI, costruiscono la risposta al loro interno definendo delle stampe virtuali in codice HTML; è quindi evidente che ogni variazione nella risposta comporta la necessità di ricompilare la servlet.

Le JSP (*Java Server Pages*) sono un tentativo di risolvere questo inconveniente in quanto non sono altro che pagine HTML che incorporano codice Java che viene sempre eseguito sul lato server. In pratica possiamo affermare che una JSP sta ad una servlet come una pagina PHP sta ad un programma CGI.

Più in dettaglio le JSP rappresentano un esempio di tecnologia Java in grado di integrare in uno stesso file codice HTML, componenti riutilizzabili come i Javabeans, codice Java e script "Java-like".

Una pagina JSP appare come una normale pagina HTML contenente anche tag JSP attraverso i quali si possono definire singole operazioni (ad esempio chiamate al codice esterno di un componente riusabile Javabeans) o blocchi di codice Java, chiamati *SCRIPLET*, che vengono compilati ed eseguiti quando si accede al documento.

Una pagina JSP viene eseguita da un "JSP engine" installato sul WEB server, che non fa altro che creare dinamicamente ed in modo trasparente la servlet corrispondente. Questo permette di conservare i vantaggi delle servlet e di superare i problemi dovuti alla loro eccessiva rigidità nei confronti delle modifiche.

Ovviamente i lati positivi di una soluzione sono sempre, almeno in parte, bilanciati da dei lati negativi: le JSP impogono la presenza sul server di un compilatore Java, non necessario per le servlet, e hanno una minore velocità di esecuzione.



## 4.6 WEB ad oggetti

Il protocollo HTTP con l'interfaccia CGI è abbastanza lento e inadeguato per applicazioni che prevedano forte interazione tra componenti o oggetti in esecuzione sul cliente e sul server. Come abbiamo visto, la modalità standard di interazione client/server dell'HTTP è data dalla FORM che è senz'altro insufficiente in caso di una architettura ad oggetti in cui, ad esempio, un oggetto client debba "invocare" direttamente un oggetto server. Inoltre nell'attuale architettura l'elaborazione è quasi sempre spostata sul server e ciò non contribuisce certo a bilanciare il carico elaborativo tra i nodi di una rete.

Un passo avanti importante, in questo ambito, si ha con l'introduzione nel WEB di soluzioni basate sugli oggetti ottenendo il cosiddetto "WEB ad oggetti". In questo ambiente gli oggetti client possono fare riferimento direttamente agli oggetti server e i vari componenti possono essere residenti su più nodi di elaborazione distribuendo così il carico elaborativo.

L'uso degli oggetti nella programmazione WEB contribuisce poi alla diffusione della "componentistica software", termine con il quale si indica una tecnologia tendente a portare nel campo della produzione del software alcune metodologie di tipo industriale "classico", superando i metodi artigianali preesistenti. Con essa lo sviluppo delle applicazioni si scompone in due attività distinte:

- creazione di componenti ("semilavorati" software);
- costruzione di applicazioni integrando i componenti già pronti.

Il WEB ad oggetti richiede la presenza di una infrastruttura di oggetti distribuiti come CORBA (Common Object Request Broker Architecture) che è una tecnologia aperta, disponibile per molte piattaforme, definita da un cartello di industrie informatiche (IBM, SUN, NETSCAPE ed altre) o DCOM (Distributed Common Object Model) che è la tecnologia analoga di Microsoft su cui si basano i controlli ACTIVEX.

Entrambe le tecnologie non sono legate a specifici linguaggi di programmazione: un server è in grado di mettere a disposizione servizi di un processo servente a vari client, residenti su macchine diverse, anche se realizzati con linguaggi differenti.

La comunicazione tra gli oggetti avviene solitamente in reti TCP/IP e può usare protocolli come l'HTTP oppure protocolli definiti appositamente come, per CORBA, l'IIOP (Internet InterOrb Protocol).

L'indifferenza dal linguaggio di programmazione pone la questione di come descrivere gli oggetti CORBA o DCOM in modo che i programmi possano fare riferimento ad essi qualunque sia il linguaggio in cui sono scritti. A questo scopo è stato definito un apposito linguaggio per la definizione di interfacce, l'IDL (Interface Definition Language) per CORBA e l'MIDL per DCOM.

La tendenza attuale è quella di utilizzare CORBA come infrastruttura per l'interoperabilità tra le componenti e di scrivere queste ultime utilizzando Java; in questo contesto CORBA rende trasparente la presenza della rete e Java rende universale l'implementazione degli oggetti in quanto ormai qualsiasi piattaforma possiede una propria Java Virtual Machine.

## 4.7 Conclusioni

In questo capitolo si sono introdotti una serie di argomenti, ognuno dei quali, come più volte fatto notare, avrebbe meritato ben altro spazio e approfondimento. Qui si è cercato solo di fornire una panoramica sui principali strumenti per rendere le pagine WEB dinamiche e interattive; il lettore interessato può approfondire gli aspetti che ritiene più utili e interessanti consultando i testi che trattano tali argomenti.

Essendo questi ultimi numerosissimi può essere utile dare qualche indicazione:

Per Linux e il "movimento Open Sources":

"I segreti di Red Hat Linux" di N. Barkakati - Ed. Apogeo 1999

"Open Sources" di R. Stallman, L. Torvalds e altri - Ed. Apogeo



Per Apache:

"Apache" di B. Laurie, P. Laurie - Ed. Apogeo 1997

Per il linguaggio Perl:

"Perl guida di riferimento" di E. Siever, S. Spainhern - Ed. Apogeo

Per il linguaggio SQL:

"Guida a SQL" di A. Guidi, D. Dorbolò - Ed. McGraw-Hill 1996

Per Linux, Apache, Perl, PostgreSQL:

"Appunti Linux" di D. Giacomini (ora "Appunti di informatica libera") - URI:

<http://www.allnet.it/AppuntiLinux>

Per PHP:

"PHP manual" di autori vari - URI: <http://www.php.net/manual>

Per JavaScript e DHTML:

"JavaScript la guida" di D. Flanagan - Ed. Apogeo

"JavaScript esempi di programmazione" di S. Feather - Ed. Jackson Libri

"CSS e DHTML" di D. Livingston, M. Brown - Tecniche Nuove 2000

Per la programmazione ad oggetti:

"Programmazione orientata agli oggetti" di J. Cox Brad - Ed. Addison-Wesley 1990

Per Java:

"Java 1.2 guida completa" di L. Lemay, R. Cadenhead - Ed. Apogeo 1997

"Java guida al linguaggio per scrivere pagine WEB interattive" di M. e O. Gurewich - Ed. Mondadori 1996

Per molti degli argomenti che riguardano le pagine WEB statiche e dinamiche:

"Sito di HTML.IT" - URI: <http://www.html.it>



## 5 DA HTML A XML: INTRODUZIONE

### 5.1 Limiti dell'HTML

Come abbiamo visto all'inizio del corso, l'HTML deriva dall'SGML che è un linguaggio a marcatori molto più potente, complesso e disponibile dal oltre 20 anni in varie forme e per vari scopi. Esso infatti non è solo un linguaggio di formattazione o di contrassegno ma è un "metalinguaggio" con il quale gli utenti possono definire propri linguaggi di contrassegno per oggetti molto diversi: formule matematiche, spartiti musicali, formule chimiche, ipertesti (è il caso dell'HTML).

Uno di questi linguaggi è l'XML del quale parliamo brevemente in questo capitolo.

L'SGML specifica degli identificatori di contenuto con i quali formattare il testo in modo coerente per permettere ai sistemi di gestione dei documenti di reperire facilmente le informazioni; si presta bene per la gestione di grandi quantità di dati con struttura omogenea come cataloghi, manuali, tabelle statistiche.

Siccome usa marcatori basati sul contenuto e non sul formato, è possibile cambiare agevolmente le regole di formattazione per inviare il documento a dispositivi di memorizzazione o di visualizzazione di tipo diverso.

Nonostante questi pregi l'SGML è utilizzato solo da organizzazioni di una certa mole, come l'IBM o alcuni enti statali americani, in quanto è notevolmente complesso e difficile da usare in modo corretto.

L'HTML conserva traccia delle sue "origini" soprattutto nella sintassi dei marcatori ma è indubbiamente molto più semplice e molto più limitato nelle funzionalità. Malgrado questo ha svolto un ruolo importante contribuendo in modo decisivo alla diffusione e al successo delle pagine WEB presso il grande pubblico.

I limiti più evidenti dell'HTML sono:

- la rigidità, in quanto ha un numero finito di tag;
- la scarsa coerenza, in quanto prevede sia tag che indicano la natura di un elemento, come <TITLE>, sia tag che indicano la sua rappresentazione, come <FONT>, sia, ancora, tag "ibridi", come <P ALIGN="center"> o <TABLE> usato per migliorare il layout della pagina.

E' utile soffermarsi a questo punto sulle caratteristiche che ogni elemento di un documento possiede:

- sintassi: cioè come è scritto;
- semantica: cosa significa, cosa rappresenta, in che relazione sta con gli altri elementi;
- rappresentazione: come appare sullo schermo o in stampa;
- comportamento: come reagisce all'interazione con l'utente (vale per gli elementi dinamici).

La maggior parte dei tag dell'HTML ha sintassi, semantica, rappresentazione e comportamento predefiniti e quindi i browser dovrebbero essere in grado di trattarli uniformemente; il condizionale dipende dal fatto, più volte evidenziato, che ogni browser in effetti fa "di testa sua".

Con il passare degli anni e con la sempre maggiore diffusione, l'HTML si è via via discostato dalla sua natura iniziale di strumento per la definizione della struttura dei documenti per divenire un linguaggio di formattazione, spostando quindi l'accento dalla semantica alla rappresentazione degli elementi.

Questo è avvenuto malgrado l'introduzione dei CSS che permettono di definire la formattazione della pagina, liberando da questo onere i tag HTML e restituendoli alla loro funzione originale di descrittori del contenuto.

Abbiamo quindi, da una parte un linguaggio ricco di potenzialità, molto rigoroso e versatile ma eccessivamente complesso come l'SGML, dall'altra un linguaggio molto semplice e di larga diffusione ma molto rigido e incoerente come l'HTML; in questo contesto si inserisce il linguaggio XML.



## 5.2 Genesi e natura dell'XML

XML (*eXtensible Markup Language*) è un linguaggio a marcatori derivato dall'SGML con lo scopo dichiarato di conservarne l'80% delle potenzialità con solo il 20% della complessità.

E' stato creato da un gruppo di lavoro del W3C coordinato da Tim Berners-Lee e si propone come base per gli sviluppi futuri del WEB.

Con XML è possibile definire nuovi marcatori (questo è il motivo del termine extensible) e specificare separatamente sintassi, semantica, rappresentazione e comportamento di ogni tag.

Grazie alla possibilità di definire nuovi marcatori si può affermare che anche l'XML, come l'SGML è un metalinguaggio attraverso il quale definire linguaggi da usare in vari ambiti; ad esempio già esistono, basati su XML: CML (*Chemical Markup Language*) per la chimica, MathML (*Mathematical Markup Language*), WML (*Wireless Markup Language*) per la navigazione con i telefonini e tanti altri.

La differenza sostanziale tra HTML e XML è che quest'ultimo si usa per descrivere il significato dei dati e non il loro aspetto; questo è un grosso passo avanti che, insieme alla flessibilità e versatilità del linguaggio, apre prospettive di utilizzo al di là della rappresentazione dei dati nelle reti e nel WEB. L'XML si propone infatti come possibile formato universale per la rappresentazione dei dati superando i cronici problemi di incompatibilità tra formati generati con applicazioni diverse.

Si deve infine notare come l'XML non sia stato concepito per sostituire HTML, ma per integrarlo ed estenderlo, tanto è vero che quest'ultimo può essere reinterpretato nell'architettura XML dando origine ad un nuovo linguaggio: l'XHTML (*eXtensible HTML*).

## 5.3 Documenti XML ben formati e validi

Un documento XML è un semplice file testuale, realizzabile quindi con un qualsiasi editor, contenente testo e marcatori. Questi ultimi sono racchiusi fra i simboli "<" e ">" come in HTML; a differenza che in quest'ultimo, però, qui ci sono regole sintattiche molto rigide:

- ogni tag deve essere necessariamente chiuso con il relativo tag di chiusura;
- non è possibile avere annidamenti con tag sovrapposti; ciò significa che la seguente porzione di codice:

```
<U> testo sottolineato<B> e grassetto </U></B>
```

che in HTML è accettata, anche se scorretta, in XML è del tutto vietata;

- i valori degli attributi devono essere sempre racchiusi tra virgolette;
- deve essere sempre presente almeno un elemento e fra gli elementi uno ed uno solo ha un ruolo speciale ed è chiamato radice; gli altri elementi sono tutti "figli", "nipoti" e comunque discendenti della radice secondo una struttura gerarchica.

Un documento che rispetti queste regole si dice "**ben formato**" o "**conforme**".

Vediamo un primo esempio di documento XML (biblioteca.xml) in cui si definiscono dati relativi ad una biblioteca usando dei tag definiti liberamente a questo scopo:

```
<?xml version="1.0" standalone="yes" ?>
<BIBLIOTECA>
  <LIBRO>
    <TITOLO>XML LE BASI</TITOLO>
    <AUTORE>S. ST. LAURENT</AUTORE>
    <CASAED>TECNICHE NUOVE</CASAED>
    <ARGOMENTO>WEB - XML</ARGOMENTO>
```





```
<ANNOED>1999</ANNOED>
</LIBRO>
<LIBRO>
  <TITOLO>PRIMI PASSI CON LINUX</TITOLO>
  <AUTORE>P. D'IGNAZIO</AUTORE>
  <CASAED>INFOMEDIA</CASAED>
  <ARGOMENTO>S.O. - LINUX</ARGOMENTO>
  <ANNOED></ANNOED>
</LIBRO>
</BIBLIOTECA>
```

L'indentatura, come al solito, ha il solo scopo di aumentare la leggibilità del sorgente. Si deve notare che il linguaggio è "case sensitive", quindi `<LIBRO>` e `<libro>` sono tag diversi.

Nel caso di elementi vuoti si può usare una notazione abbreviata:

```
<ANNOED></ANNOED>
```

può essere sostituito con

```
<ANNOED/>
```

Nell'esempio l'elemento radice è `<BIBLIOTECA>` mentre la prima riga è un po' particolare in quanto rappresenta una istruzione di elaborazione XML; tali istruzioni iniziano con i caratteri "`<?>`" e terminano con "`?>`". Questa istruzione presenta due attributi: *version* il cui significato è ovvio e *standalone* il cui significato verrà chiarito più avanti.

Le regole elencate in precedenza forniscono solo un primo livello di validazione; un secondo livello si ottiene grazie alla DTD (*Document Type Definition*) che è un testo che descrive le regole sintattiche per il documento XML e può essere contenuto in quest'ultimo oppure, più frequentemente, risiedere in un file esterno.

La seconda alternativa è ovviamente da preferire in caso di più documenti definiti con una stessa DTD in quanto permette di scrivere quest'ultima una volta sola.

Un documento che rispetti le regole sintattiche di una DTD si dice "**valido**".

L'esempio "biblioteca.xml" deve essere così modificato nella parte iniziale:

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE BIBLIOTECA [
  <!ELEMENT BIBLIOTECA (LIBRO+)>
  <!ELEMENT LIBRO (TITOLO, AUTORE*, CASAED, ARGOMENTO, ANNOED?)>
  <!ELEMENT TITOLO (#PCDATA)>
  <!ELEMENT AUTORE (#PCDATA)>
  <!ELEMENT CASAED (#PCDATA)>
  <!ELEMENT ARGOMENTO (#PCDATA)>
  <!ELEMENT ANNOED (#PCDATA)>
]>
<BIBLIOTECA>
  <LIBRO>
.....
```

La DTD può essere esterna e residente, ad esempio, nel file "biblioteca.dtd" seguente:

```
<?xml version="1.0" ?>
<!ELEMENT BIBLIOTECA (LIBRO+)>
```



```
<!/ELEMENT LIBRO (TITOLO, AUTORE*, CASAED, ARGOMENTO, ANNOED?)>
<!/ELEMENT TITOLO (#PCDATA)>
<!/ELEMENT AUTORE (#PCDATA)>
<!/ELEMENT CASAED (#PCDATA)>
<!/ELEMENT ARGOMENTO (#PCDATA)>
<!/ELEMENT ANNOED (#PCDATA)>
```

Allora il documento "biblioteca.xml" assumerebbe questo aspetto:

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE BIBLIOTECA SYSTEM "biblioteca.dtd">
<BIBLIOTECA>
  <LIBRO>
.....
```

A questo punto dovrebbe essere chiaro il significato dell'attributo standalone: vale "no" quando almeno una definizione DTD è esterna al documento.

La parola chiave SYSTEM si usa per DTD definite localmente; in certi casi è utile usare DTD pubbliche già definite con questa sintassi:

```
<!DOCTYPE nome PUBLIC "nome DTD" "url DTD">
```

Le linee contenute nella DTD si chiamano "dichiarative", iniziano con i simboli "<!" seguiti dal tipo di dichiarativa e terminano con ">".

Il nome scritto nella dichiarativa !DOCTYPE (che ricordo essere presente anche nei sorgenti HTML) deve coincidere con la radice del documento XML a cui la DTD si riferisce; tra le parentesi quadrate è racchiusa la definizione del documento.

Il tipo !ELEMENT permette di specificare il tipo di elemento; la sua sintassi è:

```
<!/ELEMENT nome contenuto>
```

Nel nostro esempio il primo elemento ha nome BIBLIOTECA e contiene uno o più di uno (questo è il significato del simbolo "+") elementi LIBRO.

A sua volta LIBRO contiene una sequenza di elementi (la sequenza si indica elencando gli oggetti separati da virgola) TITOLO, AUTORE, CASAED, ARGOMENTO, ANNOED. In caso si dovesse invece avere la scelta tra una lista di elementi la sintassi da usare sarebbe (el1 | el2 | el3 ....).

AUTORE è opzionale e può esserci una o più volte (è il significato del simbolo "\*"); PREZZO può esserci zero o una volta (è il significato del simbolo "?"). Gli altri elementi devono tutti comparire una ed una sola volta.

La parola chiave #PCDATA indica che quell'elemento è un elemento di testo.

Un altro tipo di dichiarativa è !ATTLIST che permette di indicare gli attributi di un elemento, con questa sintassi:

```
<!ATTLIST nome terna*>
```

dove "terna" è composta da *nome-attributo tipo-attributo valore-default*

ad esempio:

```
<!ATTLIST ARGOMENTO
      id          ID          #REQUIRED
      nome-fig    CDATA      #IMPLIED
```



visible (yes | no) "yes" >

ID indica l'identificativo del tag e permette di identificarlo univocamente nel documento, CDATA una stringa di testo, #REQUIRED impone che l'attributo sia specificato, #IMPLIED indica che l'attributo non è obbligatorio.

Un altro valore possibile è #FIXED che indica un attributo con valore fisso.

Il terzo tipo di dichiarativa è quello per le entità, !ENTITY che può essere di due tipi: generale o di parametro.

Le entità generali sono definite nelle DTD ma si usano poi nei documenti XML; la loro sintassi è:

*<!ENTITY nome definizione>*

ad esempio per definire il carattere "&" si usa:

*<!ENTITY amp "&#38">*

oppure per definire un'entità associata ad un testo ripetitivo da inserire più volte in un documento:

*<!ENTITY sost "Questo è il testo sostitutivo">*

Il riferimento alle entità si effettua antepoendo il simbolo "&" e posponendo il simbolo ";".

Quindi se in un documento si vuole inserire il testo dell'esempio sopra illustrato si scrive:

*&sost;*

Le entità di parametro si usano esclusivamente nelle DTD esterne; la loro sintassi è:

*<!ENTITY % nome definizione>*

Queste entità vengono di solito usate per fare riferimento a definizioni di dati contenute in file esterni; *definizione* in questo caso corrisponde ad un url.

Non approfondiamo ulteriormente questi argomenti; maggiori dettagli sulle DTD si possono reperire liberamente sul sito del consorzio W3C: <http://www.w3.org>.

## 5.4 Parser ed applicazioni XML

Come abbiamo visto un documento XML contiene solo la definizione di un insieme di dati ottenuta grazie ad un set arbitrario di marcatori. Sorgono a questo punto due problemi:

- 1 - il documento deve essere analizzato e validato;
- 2 - i dati devono in qualche modo essere gestiti (ad esempio visualizzati con un browser).

Gli stessi problemi in verità esistono anche per i documenti HTML che però contengono tag appartenenti ad un insieme rigido e prefissato; quindi è bastato costruire i modo opportuno i browser "istruendoli" su come interpretare e gestire i vari tag ed i problemi sono stati risolti.

Nel caso dell'XML le cose sono un po' più complicate; la questione della validazione viene affrontata dai PARSER XML che svolgono le seguenti operazioni:

- analizzano il documento verificando che sia conforme e, nel caso sia specificata una DTD, valido;



- lo suddividono nelle sue componenti generando una qualche forma di output specifico per la piattaforma di esecuzione.

Esistono due tipi di parser:

- i parser DOM (*Document Object Model*), che forniscono in output una struttura gerarchica ad albero che rispecchia la struttura del documento XML; tale struttura è indipendente dalla piattaforma ed è adatta per applicazioni interattive perché viene sempre mantenuta interamente in memoria centrale;
- i parser SAX, che leggono il documento XML e generano eventi corrispondenti alla sua struttura; richiedono un uso più limitato della memoria ma impediscono di tornare indietro nell'analisi degli elementi XML. Sono preferiti nel caso di filtraggio dei dati o di applicazioni server-side nelle quali non sia richiesta la rappresentazione in memoria dei dati.

Attualmente tutti i browser più recenti sono in grado di effettuare il parsing di un documento XML.

Il problema della gestione dei dati viene invece affrontato dalle applicazioni XML che si occupano di manipolare le informazioni ricevute dal parser trasformandole in elementi di varia natura adatti ad essere visualizzati, stampati, memorizzati o inviati ad altri dispositivi di output.

Nel caso il documento XML debba essere solo visualizzato con un browser è possibile ricorrere semplicemente ai CSS, a patto che il browser sia almeno in grado di fare il parsing.

Riprendendo l'esempio "biblioteca.xml" vediamo un possibile foglio di stile, "biblioteca.css", che permette la sua visualizzazione sia con MOZILLA 0.9 sia con INTERNET EXPLORER 5.5:

```
/* CSS per esempio biblioteca.xml */
BIBLIOTECA { background-color: white}
LIBRO { display: block;
        padding-bottom: 0.4in}
TITOLO { display: block;
        text-align: center;
        color: blue;
        padding: 0.2in;
        font-style: italic;
        font-weight: bold;
        font-size: 18pt}
AUTORE, CASAED, ARGOMENTO, ANNOED { display: block;
        text-align: left;
        text-indent: 3.8in;
        color: black;
        font-size: 14pt}
```

Per usarlo occorre aggiungere a "biblioteca.xml", come seconda linea:

```
<?xml-stylesheet type="text/css" href="biblioteca.css" ?>
```

Il risultato che si ottiene aprendo "biblioteca.xml" con MOZILLA è il seguente:



### XML LE BASI

S. ST. LAURENT  
TECNICHE NUOVE  
WEB - XML  
1999

### PRIMI PASSI CON LINUX

P. D'IGNAZIO  
INFOMEDIA  
S.O. - LINUX

Con i fogli CSS si può comunque solo modificare il formato degli elementi XML ed agire solo sui singoli elementi. Se si vuole invece manipolare in modo più consistente un documento XML, ad esempio riordinando gli elementi o nascondendone alcuni, si deve ricorrere ai fogli XSL.

L'XSL (*eXtensible Style Language*) comprende sia un linguaggio di trasformazione, l'XSLT (*XSL Transformation*), che uno di formattazione, l'XSL:FO (*XSL: Formatting Object*), entrambi definiti usando XML.

La trattazione sull'uso di questi linguaggi nonché dei linguaggi XLink e XPointer per il collegamento tra documenti XML e dell'XMLSchema definito anch'esso in XML e alternativo alle DTD, esula dagli scopi di questo corso; si rimanda, come al solito, all'ampia letteratura disponibile sull'argomento.



## 6 BIBLIOGRAFIA

"Il manuale di HTML 4.0" di S.E. Mack, J. Platt - Ed. Jakson Libri 1998

"Professione Internet" di autori vari - Ed. McGraw-Hill - La Repubblica 1998

"Linux e programmazione WEB" di M. Sciabarrà - Ed. McGraw-Hill 1999

"XML Le basi" di S. ST. Laurent - Tecniche Nuove 1999

"Appunti Linux" (ora "Appunti di informatica libera") - URI: <http://www.allnet.it/AppuntiLinux>

Articoli vari da riviste tra cui:

"Inter.net" - Ed. Systems comunicazioni S.r.l. - <http://www.interpontonet.it>

"Linux & C." - Ed. Piscopo S.r.l. - <http://www.oltrelinux.com>

"Linux Magazine" Ed. Edizioni Master - <http://www.edmaster.it>

Articoli vari da riviste elettroniche tra cui:

"Pluto Journal" - <http://www.pluto.linux.it>

"Linux Gazete ed. italiana" - <http://linux.cassino.edu/lgei>