Appunti su XML

I testi sono stati elaborati prendendo spunto, oltre che dalle fonti bibliografiche e dai link citati in chiusura, dagli appunti raccolti da SABRINA AURICCHIO, ILARIA BARBANTI, SILVIA MAZZINI E CHIARA VENINATA durante il corso di formazione *"Labirinti – archivi, labirinti della memoria"*, 2001-2002.
Ad una prima sezione introduttiva, dedicata alla trattazione di alcuni concetti generali, segue una breve esposizione su XML, strumentale allo svolgimento delle esercitazioni.

ALCUNI CONCETTI

Standard

Per far fronte al proliferare della comunicazione sul Web, e per ovviare al rischio di una 'babele telematica' con effetti deleteri sul piano della portabilità dei documenti[1], si è imposta la necessità di regolarne i mezzi mediante *standard*. Per standard si intende qualsiasi insieme di norme di progettazione e uso relativo a una particolare tecnologia, emesse da un ente istituzionale nazionale o internazionale, al fine di favorire la condivisione di oggetti, metodi e processi finalizzati all'interoperabilità degli stessi all'interno di una comunità. In tal modo viene garantita sia la portabilità tra hardware e software diversi (*portabilità interpiattaforma*), sia la stabilità e riusabilità delle risorse stesse.

La tendenza attuale della codifica delle risorse elettroniche sulla rete punta a costruire ambienti standard in grado di permettere la costruzione di applicazioni portabili a prescindere dal sistema operativo o dal tool di sviluppo.

Lo scambio del contenuto informativo delle risorse in ambiente elettronico deve perciò avvenire secondo una logica di gestione il più possibile funzionale all'accesso e alla condivisione delle informazioni stesse, mediante sistemi di codifica che rispettino requisiti come:

- l'esplicitazione della struttura logica che sottende il contenuto e la forma con cui esso viene rappresentato;
- l'universalità dell'accesso alle risorse.

Per tali ragioni è preferibile che qualsiasi sistema di codifica delle risorse in ambiente elettronico adotti un formato standard.

Affinché uno standard di rappresentazione dell'informazione sia effettivamente portabile deve poter garantire sostanzialmente:

- *l'indipendenza dall'hardware*, ovvero da una particolare architettura elaborativa (processore), da un particolare supporto digitale (disco magnetico, disco ottico, etc.), o da un particolare dispositivo o sistema di output (video, stampa);
- *l'indipendenza dai software*, siano essi *sistemi operativi* o applicazioni deputate alla creazione, analisi, manipolazione e visualizzazione di testi elettronici, nella maggior parte dei casi incompatibili tra loro e troppo rapidamente obsoleti.

L'ente internazionale deputato alla definizione di *standard pubblici* formali è la ISO – *International Standardization Organization*, composta da membri dei vari enti di standardizzazione nazionali. Esistono anche casi in cui una data comunità di utenti, dotata di strutture organizzative più o meno formalizzate, può decidere di sviluppare o adottare norme comuni per la rappresentazione ed il trattamento dell'informazione, una sorta di standard informale ad uso interno[2]. Infine, nel mondo dell'informatica molti sistemi ed applicativi si impongono *de facto* per la penetrazione commerciale e poi vengono eventualmente adottati formalmente da un ente istituzionale (viceversa non tutti gli standard pubblici riescono ad imporsi o a scalzare uno standard di fatto).

Tra le tipologie di standard a cui si è accennato, lo standard pubblico, oltre ad essere riutilizzabile, offre anche il vantaggio della disponibilità pubblica delle sue specifiche tecniche. L'adozione di standard nella creazione di risorse digitali, pertanto, riduce notevolmente i rischi di "deperibilità" determinati dai formati commerciali (infatti l'evoluzione del software non modifica il formato dei dati, ma solo le procedure di elaborazione), e rende possibile lo sviluppo di software a distribuzione gratuita.

---

[1] Per "portabilità" si intende la possibilità di leggere i documenti Web con qualsiasi browser e su qualsiasi piattaforma.

[2] Questo è avvenuto, ad esempio, per la comunità internazionale degli studiosi di informatica umanistica, che ha dato vita alle norme di codifica testuale della *Text Encoding Initiative*, di cui parleremo più avanti. Le norme TEI a loro volta sono basate su un linguaggio di codifica che è uno standard formale ISO, lo *Standard Generalized Markup Language*.

Approfondimento – W3C (World Wide Web Consortium)

Accanto all'ISO riveste notevole importanza il *World Wide Web Consortium* (W3C), organizzazione *no profit* ufficialmente deputata allo sviluppo degli standard tecnologici per il Web che raccoglie centinaia di aziende, enti e centri di ricerca coinvolti più o meno direttamente nel settore. Il lavoro si articola per commissioni e gruppi di lavoro, che producono bozze tecniche di lavoro (*working drafts*). Ogni proposta viene poi sottoposta ad un processo di verifica e di revisione, finché non viene approvata dal consiglio generale e diventa una 'raccomandazione' (*recommendation*), alla quale è possibile far riferimento per sviluppare software. In questi ultimi anni il W3C ha prodotto una serie di specifiche divenute, o in procinto di divenire, standard ufficiali su Internet. Tutti i materiali prodotti dal W3C sono di pubblico dominio, e vengono pubblicati sul sito Web del consorzio: http://www.w3c.org (vedia anche http://www.w3c.it).
Tra le attività del W3C si citano le specifiche relative ai linguaggi di marcatura di tipo dichiarativo SGML, HTML e XML.

Linguaggi del Web

World Wide Web definisce formati specifici per codificare i documenti che vi vengono immessi e distribuiti.

I documenti che costituiscono la rete ipertestuale del Web sono principalmente documenti testuali, ai quali possono essere associati oggetti grafici (fissi o animati) e in taluni casi moduli software.

In generale la struttura, i contenuti e l'aspetto di una pagina Web visualizzata da un dato *user agent* sono definiti interamente nel documento testuale che ne costituisce l'oggetto principale. Per realizzare tutto ciò vengono creati linguaggi di codifica, tra i quali vi sono quelli di codifica dei testi.

Markup – linguaggio di codifica dei testi

Un testo può essere suddiviso in struttura concettuale (semantica, contenuto) e rappresentazione grafica. Una semplice analisi ci permette di evidenziare in esso un'organizzazione del discorso in macrounità, titoli e sottotitoli, ma anche la modalità con cui si identifica ognuno di questi elementi. Per rappresentare su supporto informatico tutte le caratteristiche grafiche e strutturali di un documento vanno adottati formalismi più complessi. Tra questi vi sono i cosiddetti *markup language*, linguaggi di marcatura.

L'espressione *markup* deriva dalla analogia tra questi linguaggi e le annotazioni inserite da tempo da autori, curatori editoriali e correttori nei manoscritti e nella bozze di stampa di un testo al fine di indicare correzioni e trattamenti editoriali, chiamate in inglese *mark up*. In modo simile, i linguaggi di codifica testuale sono costituiti da un insieme di istruzioni che servono a descrivere la struttura, la composizione e l'impaginazione del documento. I marcatori sono sequenze di normali caratteri ASCII (*American Standard Code for Information Interchange*, standard di codifica dei caratteri), e vengono introdotti, secondo una determinata sintassi, all'interno del documento, accanto alla porzione di testo cui si riferiscono.

I linguaggi di markup si dividono in:

- *linguaggi procedurali*, definiti anche *specific markup language*
- *linguaggi dichiarativi* o descrittivi, detti anche *generic markup language*.

Linguaggio procedurale

I linguaggi procedurali contengono una serie di istruzioni operative che indicano la struttura tipografica della pagina, le spaziature, l'interlineatura, i caratteri usati, in altre parole lo stile e la

formattazione del documento(*layout*). Questo tipo di linguaggi sono detti *procedurali* in quanto istruiscono un programma su come trattare la sequenza di caratteri al momento della stampa. Un esempio di linguaggio procedurale è il *Rich Text Format*, un formato sviluppato dalla Microsoft per consentire lo scambio interpiattaforma di documenti formattati.

Linguaggio dichiarativo

Nei linguaggi dichiarativi o descrittivi le istruzioni utilizzate per assegnare ogni porzione di testo ad una certa tipologia di caratteristiche testuali sono i marcatori (tag); essi permettono di descrivere la struttura logica di un documento, il modo con cui viene organizzato il contenuto, sistemando l'insieme testuale in una sorta di gerarchia al cui vertice (*root*) si trova l'elemento (es.: libro) che contiene tutti gli altri (titoli, paragrafi, note, ecc.). esempi di linguaggi dichiarativi sono HTML, SGML e XML.
Tra i vantaggi che derivano dallo sviluppo di applicazioni di codifica testuale mediante un linguaggio di tipo dichiarativo, si indicano i seguenti:

- In fase di pubblicazione un documento codificato in modo dichiarativo può essere visualizzato su diversi media e supporti di pubblicazione (stampa, disco ottico, rete telematica) senza che per questo vi sia la necessità di modificare la codifica effettuata.
- Se il linguaggio di codifica risponde a determinati criteri di standardizzazione, il testo è portabile virtualmente su qualsiasi piattaforma hardware e software.
- I sistemi di codifica dichiarativa sono in grado di fornire informazioni anche per la gestione e la visualizzazione di strutture complesse (riferimenti incrociati, collegamenti ipertestuali, ecc.), senza bisogno di specificarle attraverso comandi e dispositivi di un software proprietario.

Approfondimento – TAG

Nel tentativo di risolvere i problemi interni di pubblicazione di una serie di informazioni, nel 1969 l'IBM sviluppò il Generalized Markup Language (GML). Da allora questo sistema è stato acquisito da tutti i linguaggi di codifica testuale di tipo dichiarativo per indicare delle "etichette" o elementi con cui si codifica il testo, operazione che sta alla base di ogni processo di marcatura. I tags sono riconoscibili poiché racchiusi in parentesi uncinate: <> in apertura e </> in chiusura. Al loro interno vi sono il nome dell'elemento e eventuali suoi attributi, presenti solo nel tag di apertura:

Es.: <lettera data="09/05/01">.....</lettera>
Lettera= nome del tag
Data= nome dell'attributo
09/05/01= valore dell'attributo

Un tipo particolare di tag è il tag vuoto (*empty*), che non corrisponde a nessuna parte di testo del documento, non individua nessun testo ma marca alcune indicazioni come
 in HTML, a capo, o ancora tag che servono a spiegare il testo stesso, una sorta di commento semantico inerente il contenuto del testo marcato.

SGML – Structured Generalized Markup Language

Il testimone più illustre dei linguaggi di markup dichiarativi è lo *Standard Generalized Markup Language* (SGML). Ideato da Charles Goldfarb e sviluppato in un arco di tempo che va dal 1978 al 1986, anno di pubblicazione dello standard, SGML è divenuto lo standard ufficiale adottato dall'ISO per la creazione e l'interscambio di documenti elettronici[3].

[3] Il testo ufficiale dello standard ISO, commentato dallo stesso inventore del linguaggio, è nel fondamentale C.F. Goldfarb, *The SGML Handbook*, Oxford University Press, Oxford 1990. Manuali introduttivi di buon livello sono: E. van Herwijnen, *Practical SGML*, Kluwer Academic Publishers, Boston-Dordrecht-London 1994, II ed.; M. Bryan,

L’importanza di un linguaggio di codifica testuale, come SGML (e, come vedremo più avanti, XML), è analoga a quella dei metadati per risorse informative. Con il termine *metadati* (dati sui dati) si indica infatti l'insieme di dati estrapolabili dalla risorsa informativa poiché capaci di descriverla. Un classico esempio è costituito dai riferimenti bibliografici di una pubblicazione (titolo, autore, editore, ecc.). Essi in primo luogo identificano una pubblicazione, ed infatti sono collocati sulla copertina e sul frontespizio di un libro; poi costituiscono il contenuto delle schede nei cataloghi bibliotecari, e dunque sono necessari al reperimento del libro nel 'sistema informativo' biblioteca.
Ne consegue allora che SGML, dal punto di vista dell’esplicitazione del contenuto informativo, è definibile come *metalinguaggio*, ossia linguaggio che contiene dati sui dati, che in questo caso sono i documenti stessi. Un tag SGML introduce così ad una parte testuale individuata come significativa della struttura logica che viene rispettata nell’insieme dell’impianto semantico della risorsa.
Esso prescrive precise regole sintattiche per definire un set di marcatori e le loro relazioni, al fine di costruire un linguaggio di marcatura specifico per il documento sottoposto alla codifica.
Un’applicazione SGML descrive la struttura logica di una classe di documenti, non la loro forma fisica, dichiarando esplicitamente gli elementi che la costituiscono (ad esempio: *capitolo*, *titolo*, *paragrafo*, *nota*, *citazione*, ecc.) con i tag (<capitolo>, <titolo>, <paragrafo>, <nota>, <citazione>), e le relazioni *gerarchiche* che intercorrono tra questi. Per questo un documento SGML può essere concepito come una struttura ad albero.
Le dichiarazioni degli elementi e delle gerarchie sono contenute in un documento, denominata *Document Type Definition* (DTD), che costituisce una sorta di grammatica dei documenti che ad essa si riferiscono e rispetto alla quale debbono essere convalidati.
Una volta definito un determinato linguaggio, esso può essere utilizzato per rappresentare infiniti documenti in base ad una sintassi rigorosa. A ciascun elemento corrisponde una coppia di marcatori, nel senso che ogni elemento viene identificato da un marcatore iniziale e uno finale, a meno che non sia un elemento vuoto. Un testo codificato dunque ha il seguente aspetto[4]:

```
<text>
<front><titlepage>
<docauthor>Corrado Alvaro</docauthor>
<doctitle><titlepart type="main">L'uomo nel labirinto</titlepart>
<titlepart>in Il mare</titlepart></doctitle>
<docimprint>
<pubplace>Milano</pubplace><publisher>A. Mondadori</publisher><docdate>1932</docdate>
</docimprint>
</titlepage>
</front>
</text>
```

La codifica SGML, oltre alla sua potenza espressiva, offre una serie di vantaggi dal punto di vista del trattamento informatico dei documenti:

- un documento SGML è facilmente portabile su ogni tipo di computer e di sistema operativo, senza dovere pagare i costi di dolorose conversioni tra formati spesso incompatibili;
- un testo codificato in formato SGML può essere utilizzato per scopi differenti: stampa su carta, presentazione multimediale, elaborazione con database, ecc.

SGML: An Author's Guide to the Standard Generalized Markup Language, Addison-Wesley, Wokingham-Reading-New York 1988. Moltissime informazioni, materiali e saggi su SGML sono naturalmente reperibili su Internet. Non è possibile qui dare un elenco dei siti dedicati a SGML, ma sicuramente il più completo è la *SGML/XML Web Page* di Robin Cover disponibile su WWW all'indirizzo http://www.oasis-open.org/cover/: una vera miniera che elenca praticamente tutto quello che c'è da sapere e che c'è da trovare in rete su SGML.

[4] L'esempio è tratto da *L'uomo nel labirinto* di Corrado Alvaro (Mondadori, 1932). La codifica è basata sul DTD della TEI, ed è stata curata dal CRILet (http://crilet.let.uniroma1.it).

Per queste sue caratteristiche SGML ha trovato impiego soprattutto in contesti industriali e militari, dove la gestione efficiente e sicura dei documenti tecnici ha una funzione critica. In ambito umanistico una applicazione SGML denominata *Text Encoding Iniziative* è divenuta lo standard per la codifica e l'archiviazione dei testi su supporto digitale e per la creazione di biblioteche digitali[5].
L'applicazione SGML che gode della diffusione maggiore è lo *HyperText Markup Language*; XML invece può essere considerato un suo *subset*, un sottoinsieme semplificato che ne mantiene l'impianto sintattico e grammaticale di fondo.

HTML- HyperText Markup Language

È un linguaggio di markup dichiarativo ideato dal CERN di Ginevra nel 1989 per favorire lo scambio on-line di documentazione scientifica, ma ben presto utilizzato dall'intera comunità del Web per descrivere i documenti testuali permettendo poi di associare e integrare ad essi dati multimediali. Per averne un esempio, una volta connessi a Internet, basta collocarsi su un qualsiasi punto della pagina con il mouse e cliccare con il tasto destro su "HTML":

mettere figura

e si ha questo risultato:

mettere figura

Con HTML è possibile:

- Trattare in maniera diversa i livelli dei titoli di un documento, lo stile dei caratteri (corsivo, grassetto...), i capoversi, le liste, ecc.;
- definire link ipertestuali;
- inserire i immagini come immagini e altri oggetti;
- specificare alcune strutture interattive come moduli di immissione (*Form*) attraverso cui l'utente (*Client*) può inviare comandi e informazioni al *Server* e attivare speciali procedure (ricerche su database, invio di posta elettronica e anche pagamenti attraverso carta di credito).

Lo sviluppo di HTML è stato assai proteico e piuttosto disordinato. Nella sua prima versione ufficiale, il linguaggio era semplice, ma e non prevedeva la possibilità di rappresentare fenomeni testuali e editoriali complessi. Ancora oggi è possibile scegliere tra un insieme prefissato di elementi utili per la resa dello stile grafico, non anche per la resa della struttura del documento (descrizione bibliografica, un record di database o una poesia) o di altri oggetti in esso contenuti. Di conseguenza, le specifiche di tale linguaggio sono state introdotte revisioni ed estensioni dai vari produttori di browser Web, allargando progressivamente la capacità rappresentazionale del linguaggio, ma caratterizzandolo sempre più quale assemblatore di tecnologie, piuttosto che come linguaggio vero e proprio. Ciò ha comportato anche problemi di portabilità delle applicazioni Web, perché tutte queste tecnologie non sono standard, bensì soluzioni private più o meno diffuse, che necessitano, per essere utilizzate, di software particolari o di particolari versioni di esso.

[5] Ne abbiamo ampiamente parlato nel capitolo 'Biblioteche in rete'. Per quanto riguarda la TEI, oltre alla sezione 'Academic Applications' della già menzionata *SGML/XML Web Page* (http://www.oasis-open.org/cover), rimandiamo al sito del recentemente formato *TEI Consortium* (http://www.tei.org).

XML – FONDAMENTI ED ELEMENTI BASE DEL LINGUAGGIO

XML

XML – eXtensible Markup Language – è un linguaggio di tipo dichiarativo ideato per elaborare file di testo senza avere la necessità di dipendere, per la loro codifica, da un supporto e/o da una piattaforma applicativa locale.
XML può essere considerato un *subset*, un sottoinsieme semplificato di SGML. Per superare la complessità di SGML ma anche i limiti di HTML, è stata proposta una semplificazione dello standard, mantenendone l'impianto sintattico e grammaticale di fondo in modo da facilitarne la diffusione sul Web.
Lo sviluppo di XML è iniziato alla fine del 1996, nell'ambito della *SGML Activity* del W3C, ed è standard del W3C da febbraio 1998, quando le specifiche sono divenute una raccomandazione ufficiale, con il titolo *Extensible Markup Language (XML) 1.0* (tutti i materiali relativi al progetto, documenti ufficiali, informazioni e aggiornamenti, sono pubblicati sul sito del consorzio all'indirizzo http://www.w3.org/XML, e sul sito del W3C in italiano (vedi elenco dei siti). Gli sviluppatori di XML hanno semplicemente preso le parti migliori di SGML, guidati dall'esperienza dell'HTML, e hanno prodotto qualcosa che non è meno potente di SGML, ma molto più regolare e semplice da usare.
Come SGML XML è definibile come *metalinguaggio*; inoltre fa uso di *tags* e *attributi* (della forma `name="value"`) ma, a differenza di HTML, i tag indicano porzioni di testo stabilite da chi crea il documento codificato.
Rispetto ad HTML inoltre, il nuovo metalinguaggio può vantare diversi vantaggi:

1. l'estensibilità che permette la definizione di set di tag personalizzati
2. la salvaguardia degli elementi strutturali così come definiti in un file esterno (DTD)
3. la validazione, ossia un controllo del documento che ne attesta la conformità alle regole definite nella DTD
4. la separazione tra struttura e rappresentazione (uno degli aspetti chiave) attraverso la separazione fisica dei dati che governano i due aspetti e attraverso la separazione dei linguaggi. La rappresentazione viene infatti gestita dal foglio di stile XSL.
5. la possibilità di effettuare ricerche relative al contenuto. Sino ad ora i vari motori di ricerca (*search engines*), sono sostanzialmente sistemi di ricerca full-text, che non tengono conto della struttura del documento, ma ricercano e restituiscono documenti interi grazie all'individuazione delle cosiddete "meta – informazioni". Si tratta di parole chiave inserite nel testo codificato in HTML come valore dell'attributo content del tag "meta", secondo questa modalità:

   <META content="parola-chiave">

   Va da sé che la presenza delle parole chiave è affidata ad una scelta discrezionale del produttore della pagina.
   In XML, le informazioni veicolate dai tag possono essere utilizzate da un sistema di *information retrieval* per individuare tutte le volte che un certa stringa compare all'interno di un titolo, senza dover ricorrere alla descrizione del contenuto all'interno dei tag.

La pubblicazione delle specifiche del linguaggio XML non ha esaurito l'attività di innovazione dell'architettura Web. Infatti, intorno al progetto XML sono stati sviluppati o sono in via di sviluppo una serie di standard ulteriori che coprono altri aspetti, non meno importanti, del suo funzionamento. Per tali specifiche si rimanda alla consultazione sul sito del W3C.

Perché XML?

XML può essere utilizzato come formato di scambio di dati, trovando spazio non solo nell'area del commercio elettronico, ma anche nell'ambito delle discipline umanistiche, nello sforzo di rendere fruibile l'enorme quantità di patrimonio documentario sul Web.
Il settore archivistico può trarre numerosi vantaggi dall'uso di XML, tra cui:
- la possibilità di elaborare strumenti descrittivi normalizzati, basati su un modello dati comune
- la garanzia di raccogliere, ordinare e gestire i dati, ma anche la possibilità di conservarli a lungo periodo e migrarli senza perdita di informazioni
- la facilità di recuperare i dati agevolandone anche lo scambio su una portabilità interpiattaforma.

È allora pensabile e auspicabile codificare, ad esempio, gli strumenti che rendono possibile l'accesso alle fonti (inventari d'archivio, cataloghi bibliografici, ecc.), o anche i documenti stessi, associando all'immagine digitale dell'originale la codifica in XML delle parti testuali[6].

Come lavorare con XML

Esistono una serie di software applicativi che consentono di lavorare con XML. La scelta è attualmente molto vasta, e ogni programma, in genere, ha un impianto semplice, proprio grazie al fatto che si lavora su documenti testuali.
Nel corso delle esercitazioni verrà utilizzato uno speciale editor, XML Spy, che permette di costruire un documento XML nel rispetto delle regole formali di grammatica e sintassi proprie del linguaggio, senza ricorrere alla costruzione di un sistema di presentazione dei dati così come stabilito dal programma stesso. XML Spy permette, tra l'altro, di importare documenti e dati raccolti su database esterni.
XML Spy si compone di:
- un *parser* che esegue il controllo semantico, ricostruendo la struttura gerarchica del documento a partire dalla sua marcatura, e gestisce gli errori. L'editor impedisce di fare errori formali, nel momento dell'inserimento dei tag, e mette automaticamente il prologo qualora non ci sia.
- un *processor* che utilizza un altro file (foglio di stile XSL) in cui è definita la formattazione dei vari tag per visualizzare il documento.

XML: Well-Formed e Valid Document (qualche regola di sintassi)

Un documento XML si compone di tre elementi principali:
1. DTD, modello dati che valida l'istanza di documento;
2. l'istanza di documento, ossia uno o più documenti testuali che rispetta il modello dati implementato o validato dalla DTD;
3. Foglio di stile XSL per la rappresentazione del documento e la sua visualizzazione

Il documento XML deve essere conforme a determinate regole che garantiscono la correttezza della sintassi interna. Per tali ragioni il Parser è deputato al controllo dell'istanza di documento, valutando la conformità del documento
- alla DTD di riferimento

[6] Si veda, a tal proposito, l'elenco dei link sui progetti che prevedono l'applicazione di XML, allegato al materiale delle esercitazioni.

- alle regole della sintassi generale di XML, nel caso in cui non sia associata una DTD

Nel primo caso il documento XML è *valid*, nel secondo *well-formed*. Va da sé che nel caso del documento valido è assolutamente necessario scrivere anche un secondo file, appunto il DTD.
Le regole di sintassi generali di XML sono le seguenti:

- i tag XML sono *case – sensitive*, ossia trattano in maniera differente il carattere maiuscolo e minuscolo
- non è possibile identificare il nome di un elemento avente spazi bianchi (*white space*): non si può scrivere <**inizio capitolo**>, bensì <**inizio.capitolo**>;
- i tag XML non possono sovrapporsi. **<a>** non è permesso. Se si inizia un tag **** dentro un **<a>**, deve essere chiuso dentro il tag **<a>** come richiesto.
- non si può omettere il tag di chiusura. Se vengono scritti <**p**> e <**br**>, **</p>** e **</br>** sono obbligatori.
- se un tag non è previsto che contenga del testo può autocontenere il tag di chiusura al suo interno, cioè **
</br>** è equivalente a **
**.
- alcuni caratteri e sequenze di caratteri sono riservati, e non si possono utilizzare nei nomi di tag (%, xml, ...).

Passiamo ad analizzare le tre componenti XML sopra elencate. Parallelamente ai concetti espressi verranno fornite porzioni esemplificative tratte dagli allegati al documento. Per una trattazione più approfondita degli argomenti che verranno trattati si rimanda comunque alla consultazione di manuali che esauriscono tutte le lacune qui presenti.

1. DTD

La Document Type Definition è un insieme di regole che definisce tutti ciò che è presente in una o più istanze di documento XML (elementi, attributi, entità, come vedremo più avanti), descrivendone le relazioni interne e la struttura. In altre parole la DTD è la rappresentazione formale della struttura logica del documento: formale perché deve indicare quali sono le regole che dovranno essere rispettate al momento della marcatura. Una stessa DTD può essere condivisa da più documenti aventi la stessa tipologia.
Una DTD non è un documento XML, ossia le sue regole sintattiche non sono quelle del documento XML. In alternativa è possibile esplicitare il modello dati di un documento XML mediante un documento che rispetta le regole sintattiche del documento XML, XML Schema. In ogni caso, il documento vero e proprio della DTD o dello Schema è preceduto dal prologo

<?xml version="1.0"…?> (si veda punto 2. L'istanza di documento XML**)**

Prendendo spunto dalla DTD riportata nell'Allegato 1, vediamone alcune componenti:

a. Elementi
b. Attributi
c. Entità

- a. Elementi

Gli elementi sono le parti del documento individuate quali sue componenti essenziali. Tali componenti sono dichiarati nella DTD in questo modo:

```
Allegato 1:
```

```
<!ELEMENT lettera (intestazione, dataCronica, luogo?, salutoIniziale, corpo,
salutoFinale, firma, ps?)>
```

Gli elementi, a loro volta possono contenere altri elementi. Per individuare e isolare le parti strutturali in cui si divide il documento si inseriscono i tag all'interno del testo (si tratta, in altre parole, dell'operazione di "marcatura": vedi Allegato 4). L'elemento **root** (nel nostro caso "lettera") contiene tutto il documento.
L'articolazione degli elementi rispetta un ordine che viene deciso dal creatore e formalizzato secondo alcune regole ortografiche. Considerando l'esempio che segue, saranno citate le più importanti:

```
Allegato 1:

esempio 1:
<!ELEMENT intestazione (#PCDATA | antroponimo | qualifica | indirizzo)*>

esempio 2:
<!ELEMENT antroponimo (pseudonimo?, (nome | cognome))+>
```

- Nella **successione** degli elementi contenuti da un elemento gerarchicamente superiore (ad esempio, l'elemento *root* "lettera", che è anche *wrapper* – contenitore)
  - la parentesi tonda introduce l'elenco degli elementi, ma è anche possibile aprirne altre nel caso in cui la successione non sia obbligatoria (esempio 2)
  - la virgola "**,**" indica che gli elementi sono indicati **tutti** e in **quell'ordine** (esempio 2)
  - la barra verticale "|"indica la presenza alternativa: o un elemento o l'altro (esempio 1)
  - se un elemento è misto cioè è composto da un PCDATA (dati tipo carattere) e da un sottoelemento, il PCDATA deve essere indicato per primo
- Per dichiarare la quantità, la **ricorrenza** di ogni singolo elemento, ci si serve delle occorrenze. Esse possono essere associate ad ogni singolo elemento, oppure al di fuori delle parentesi: in questo caso l'occorrenza scelta viene adottata da tutti gli elementi contenuti nella lista. Le occorrenze sono espresse dai seguenti simboli:
  - "?" = l'elemento può essere presente da 0 : 1 volta (esempio 2);
  - "*" = l'elemento può essere presente da 0 : n volte (esempio 1);
  - "+" = l'elemento può essere presente da 1 : n volte (esempio 2).

- b. Attributi

Gli attributi permettono l'aggiunta di informazioni addizionali all'elemento, indicando una proprietà.
Già HTML prevedeva l'uso di attributi degli elementi, per esempio nel caso di:

```
<IMG align=center ...>
```

"align" altro non e che un attributo dell'elemento IMG. In linea di massima però l'uso degli attributi in HTML descrive la rappresentazione dell'elemento, e come abbiamo visto, la confusione tra i due livelli, quello della struttura e quello della rappresentazione, in XML non è possibile.
In realtà infatti gli attributi degli elementi in XML sono vere e proprie informazioni addizionali

alla struttura del documento. Nel caso dell'esempio della lettera, poiché l'elemento "antroponimo" può essere utilizzato per marcare il mittente o il destinatario, è meglio specificare in attributo a quale ruolo appartenga.
Gli attributi vengono dichiarati nell'*"attribute list" (<!ATTLIST>)* che contiene il nome dell'elemento cui gli attributi si riferiscono, il tipo di dati, la lista dei valori degli attributi stessi e il valore. Nel nostro caso, nella DTD, dopo la dichiarazione dell'elemento ANTROPONIMO si avrà un' *"attribute list"* del genere:

```
Allegato 1:

<!ATTLIST antroponimo
      ruolo (mittente | destinatario | altro) #IMPLIED
      standard (CDATA) #REQUIRED> <!-- ATTLIST elemento    nome-attributo
tipo  valore -->
```

in cui *ATTLIST* definisce la lista di attributi, *ANTROPONIMO* è il nome dell'elemento cui è riferito l'attributo, *RUOLO* e *STANDARD* sono le denominazioni degli attributi, *" mittente"* , *" destinatario"* e "*altro*" i tre valori che può assumere l'attributo a cui si riferisce.
Infine, bisogna sempre specificare determinati valori che indicano al parser come comportarsi durante il controllo di conformità:
- *#IMPLIED* indica che l'attributo può non avere valore
- *#REQUIRED* indica che l'attributo deve avere necessariamente un valore ogni volta che è usato l'elemento
- nel caso ci debba essere un valore di default, si sarebbe indicato in questo modo:

```
Allegato 1:

<!ATTLIST antroponimo
     ruolo (mittente | destinatario | altro) "altro">
```

- *#FIXED VALUE* indica che l'attributo può non avere valore ma, se lo ha, deve necessariamente essere quello di default. Tale valore viene espresso indicando direttamente il valore di default.
Nell'esempio sopra riportato è presente anche del testo contenuto tra i tag **<!--…-->**: si tratta di uno spazio ove è possibile inserire un **comment** laddove sia necessario spiegare qualcosa. Il comment è contemplato sia dalla grammatica della DTD, sia da quella XML, dunque può essere presente anche nell'istanza di documento.
I tipi di attributo possono essere diversi. I più comuni sono[7]:
1) **CDATA:** si tratta di dati di tipo carattere (testo, escluso caratteri di markup);
2) **Enumerated:** elenco di valori scritti nella DTD. Qualsiasi voce dell'elenco è un valore di attributo appropriato e deve essere usato un elemento della lista.

NOTA METODOLOGICA
Come si fa a stabilire quali sono gli elementi e gli attributi nel momento in cui si costruisce una DTD?
Le scelte sono finalizzate alla tipologia del documento, al contesto nel quale si inserisce, al perché viene adottato un processo di codifica e all'uso che se ne deve fare, cercando di prevederne l'effetto finale. Decidere quali sono gli elementi incisivi per una marcatura è un lavoro che richiede una **procedura**. Essa di solito parte dall'analisi testuale, in base alla quale si è in grado di definire una struttura

[7] Per una trattazione completa si rimanda al manuale su XML di HOLZNER, XML. Tutto & Oltre, citato tra le fonti.

semantica e individuare gli elementi. Per riconoscere gli elementi significativi si lavora redigendo una scheda che elenca gli elementi da analizzare, tra i quali fare poi una scelta, tenendo conto di due fattori:

1. il tipo di documento
2. il tipo di elaborazione che se ne deve fare

Nella scheda, per ogni elemento considerato si individuano le seguenti componenti:

1. **nome**
2. **descrizione**
3. **valutazione**

Tra quelli scelti (indicando le motivazioni, anche di quelli scartati) si costruisce la gerarchia. Di solito si parte dall'elemento *root*, per poi arrivare a definire i sottoelementi. Spesso capita che chi fa la DTD non lavora anche alla marcatura; in tal caso è auspicabile che l'intera procedura venga puntualmente documentata.

In linea generale, gli elementi corrispondono a componenti del testo, gli attributi sono proprietà degli elementi. Per introdurre qualcosa di equivalente dell'attributo vi sono diverse possibilità: si può utilizzare

1. un ulteriore elemento per scendere di più nella partizione gerarchica del testo (*granularità*): <indirizzo>Via Versilia 1, <cap>00100</cap>Roma</indirizzo>
2. un attributo che precisa una proprietà dell'elemento: <indirizzo cap="00100">Via Versilia 1, Roma</indirizzo>.

Generalmente, oltre che per il **ruolo**, si ricorre all'attributo per il **tipo** e il **valore normalizzato**. La lista degli attributi può essere data in due modi:

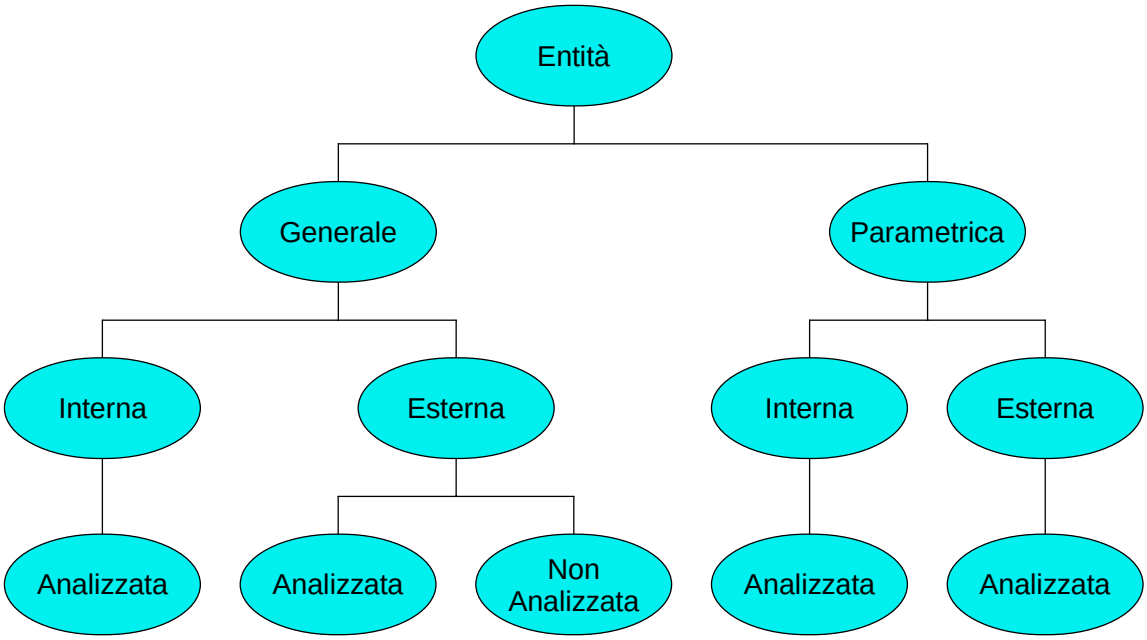
1) o subito sotto ogni elemento
2) o prima l'elenco degli elementi e poi tutti gli elementi con gli attributi

- c. Entità

Un documento XML non deve necessariamente essere composto da un solo file, ma può assemblare al suo interno pezzi diversi chiamati *"entities"*, entità, le quali permettono di creare dei riferimenti a dati esterni, ad altri documenti, o anche a porzioni di testo che vanno ripetute più volte, a patto che ci sia una dichiarazione nel DTD. Inquesto senso ciascuna istanza XML ha almeno un'entità detta entità documento (la radice dell'albero logico): essa è la prima cosa che viene analizzata dal processore XML. A questa entità documento si possono aggiungere altre che vengono richiamate in diverso modo, a seconda della loro funzione.
Questo rende possibile definire la struttura del documento nella DTD con la possibilità di riempirlo con contenuti presi dall'esterno. In questo modo anche il contenuto, oltre che la rappresentazione del documento, diventando praticamente un modulo a parte, risulta gestibile in maniera più flessibile.
Come funziona? L'entità è una sorta di "segnaposto", un nome astratto per indicare qualcos'altro: al posto dell'entità il browser inserisce ciò che debba essere visualizzato, un testo, un file, un suono. Tutte le tipologie di entità devono essere indicate sempre tutte all'inizio della DTD come una specie di glossario : **<!ENTITY nome valore>**. Il valore può essere una stringa di testo od anche un dato memorizzato esternamente. Una volta definiti nella DTD come entità delimitate da tag, una porzione di testo o un file possono essere richiamati semplicemente introducendo il riferimento all'entità .

Le entità possono essere suddivise in varie tipologie (vedi la figura 3)

Figura 3
Tipologie di entità



- Un'**entità generale** è un'entità contenuta e riusabile all'interno di un documento. Viene richiamata con il simbolo **&** che precede il nome dell'entità. Ad esempio, una tipologia di questo tipo di entità fa riferimento ad una serie di entità alfabetiche, per facilitare il compito. Eccone alcune:

```
&lt;     = <
&amp;    = &
&quot;   = “
&gt;     = >
&#169;   = ©
&apbs;   = ‘
```

- Un'**entità a parametro** è un'entità che può essere contenuta e riutilizzabile solo all'interno di una DTD. Viene richiamata antecedendo al nome dell'entità il simbolo "**%**" e può essere utilizzato solo all'interno di una DTD, negli elementi o negli attributi che seguono. Ad esempio, per definire delle abbreviazioni

```
Allegato 1:

<!ENTITY % tipo-st "CDATA #IMPLIED">
<!--CDATA = dati di tipo carattere, no tag-->
```

Quando devo richiamare un'entità ad esempio all'interno di una attribut list scriverò accanto al nome dell'attributo %tipo-st dell'esempio sopra considerato.

- Un'**entità interna** è un'entità la cui dichiarazione contiene i dati di sostituzione, ossia i dati che richiama sono riportati per intero nella DTD.

```
Allegato 1:
<!--Se nella DTD di un curriculum avrò definito come entità interna
(generale):-->

<!ENTITY dichiarazione privacy "Si autorizza il trattamento dei dati
personali ai sensi della legge 675/96">

<!--basterà poi aggiungere, nell'istanza di documento (vedi Allegato 4), il
richiamo all'entità:-->
<antroponimo ruolo="mittente">
        <nome>Massimo Stanghellini Perilli &dichiarazione.privacy;</nome>
```

- Un'**entità esterna** è un'entità la cui dichiarazione rimanda a "parametri" esterni indicati in un file esterno. Se sono dati come parametri esterni devo indicare il percorso (*path*) del file e poi specificare la sostituzione (%accent;):

```
Allegato 1:

<!ENTITY % accent SYSTEM      "accentate.txt">
% accent;
```

Si vedano, a tal proposito, gli Allegati 1 e 2.

- Un'**entità analizzata** (*parsed*) è costituita da testo XML . Il parser analizza il testo inserito con un richiamo ad una entità analizzata nello stesso modo in cui analizza il documento

- Un'**entità non analizzata** (*unparsed*) può contenere qualsiasi tipo di dati: testo non strutturato, immagini, file sonori ecc. che il parser però non è in grado di esaminare

2. L'istanza di documento (file di testo) XML

Un documento XML è costituito, dal punto di vista formale, da:
- PROLOGO: contiene solo marcatori;
- DOCUMENTO vero e proprio.

Il prologo, a sua volta, contiene due specifiche:
1. l'istruzione di elaborazione (*Processing Instruction*), inserita tra i simboli <?…?>:

```
<?XML version="1.0" standalone="yes" encoding="UTF-8"?>
```

- **<?XML .... ?>** è l'elemento che introduce il documento e lo dichiara come documento aderente alle specifiche XML.
- **"version"** definisce la versione delle specifiche di riferimento, che ovviamente, per ora non può che essere la 1.0
- **"standalone"** è l'attributo che indica se il documento è Well-Formed ("yes") o Valid ("no")
- **"encoding"** indica il tipo di codifica. Al momento quella di default è la "UTF-8", che contiene la maggior parte dei caratteri utilizzati nelle lingue dell'Europa occidentale.
- ***Encoding e standalone sono opzionali***

2. la *"Document Type Declaration"*, segnalata dai simboli <!…>. In essa si dichiara che la struttura del documento fa riferimento ad una DTD specifica, che può essere un file esterno o interno all'istanza stessa. In quest'ultimo caso, la DTD fa riferimento solo ed esclusivamente al file nel quale viene riportata.

DTD esterna:

```
<?xml version="1.0" standalone="yes" encoding="UTF-8"?>
<!DOCTYPE lettera SYSTEM "C:\WINDOWS\Desktop\cartellaXML\lettera.dtd">
```

DTD interna (inglobata dentro il documento stesso: si apre con "[" e si chiude con "]"):

```
<?xml version="1.0" standalone="yes" encoding="UTF-8"?>
<!DOCTYPE lettera [
  <!ELEMENT lettera (intestazione, dataCronica, luogo?, salutoIniziale,
corpo, salutoFinale, firma, ps?)>
ecc.
]>
```

Il file esterno (lettera.dtd) viene introdotto dalla parola chiave SYSTEM nel caso si tratti di una DTD locale o remota, comunque privata. Nel caso venga adottata una DTD pubblica (ad esempio EAD[8]) viene usata la parola PUBLIC:

```
<?xml version="1.0" standalone="yes" encoding="UTF-8"?>
<!DOCTYPE EAD PUBLIC "-//Society of American Archivist//DTD (Encoded Archival
Description(EAD)//EN" "http://linux2.hit.uib.no/xml/ead.dtd">
```

Per favorire e controllare una lettura univoca del documento XML, è stato trovato un espediente che mette in grado il parser e il processor di gestire eventuali ambiguità: i **namespaces**. Essi servono a definire il contesto a cui appartengono gli elementi, e permettono di creare e assemblare tag ambigui, ovvero con lo stesso nome, ma in riferimento a significati ed ambienti diversi utilizzando costrutti con nomi non equivoci. Si pensi per esempio a documenti di una rivista in cui la parola "titolo" a seconda del contesto può referenziare il titolo di un articolo, ma anche il ruolo del giornalista all'interno della struttura aziendale.
E' possibile creare due tag, chiamandoli entrambi *<TITOLO>*, utilizzandoli poi nel documento senza correre il rischio che il parser ed il processor li confondino. Il nome non equivoco si ottiene associando delle URI (Unified Resource Identification) agli elementi, le quali dichiarano l'ambito a cui l'elemento apartiene che fungono da ambiente per cui la seguente definizione:

```
<!—esempio non presente negli allegati→

<X xmlns:edi='http://ecommerce.org/schema'></X>
```

applica l'ambiente "edi", rappresentato dal comando "http://ecommerce.org/schema", all'elemento <X>.
La dichiarazione di Namespace si applica all'elemento all'interno del quale c'è la dichiarazione e a tutti gli altri elementi contenuti all'interno di quello a meno di nuova dichiarazione di Namespace.
Pertanto nel caso precedente tutto quello che sarebbe indentato dentro il tag *<X>* farebbe parte del Namespace *"edi"*, mentre nel caso seguente:

[8] EAD è una grammatica che permette di codificare strumenti di ricerca archivistici in formato elettronico. L'utilizzo di questa grammatica presenta evidenti vantaggi connessi alla possibilità di condividere e scambiare dati archivistici provenienti da diversi contesti, nel pieno rispetto delle particolarità locali. Rif.: http://lcweb.loc.gov/ead/.

```
<!—esempio non presente negli allegati→

<bk:book xmlns:bk='urn:loc.gov:books'
         xmlns:isbn='urn:ISBN:0-395-36341-6'>
    <bk:title>Cheaper by the Dozen</bk:title>
    <isbn:number>1568491379</isbn:number>
</bk:book>
```

all'interno del contesto <*bk:book*> si alternano due Namespace, *"bk"* e *"isbn"* ed ognuno vale fintanto che non ne è invocato un altro.

3. XSL

Il concetto di foglio di stile nasce nell'ambito delle tecnologie di *word processing* e *desktop publishing*. L'idea è quella di separare il contenuto testuale di un documento elettronico da istruzioni che ne governano l'impaginazione, le caratteristiche grafiche e la formattazione. Per fare questo è necessario suddividere il testo in blocchi etichettati e associare poi a ogni blocco uno specifico stile, che determina il modo in cui quella particolare porzione del testo viene impaginata sul video o stampata su carta. Ad esempio, ad un titolo di capitolo può essere associato uno stile diverso da quello assegnato a un titolo di paragrafo o al corpo del testo (lo stile 'titolo di capitolo' potrebbe prevedere, poniamo, un carattere di maggiori dimensioni e in grassetto, la centratura, un salto di tre righe prima dell'inizio del blocco di testo successivo; a un blocco di testo citato potrebbe invece essere assegnato uno stile che prevede un corpo lievemente minore rispetto al testo normale, e dei margini maggiori a sinistra e a destra per poterlo 'centrare' nella pagina). Per chi usa un moderno programma di scrittura come Microsoft Word o Wordperfect questo meccanismo, almeno ad un livello superficiale, dovrebbe risultare familiare.
I fogli di stile facilitano la formattazione dei documenti, permettono di uniformare lo stile di diversi testi dello stesso tipo, e semplificano la manutenzione degli archivi testuali. Sarà così sufficiente, ogni qualvolta si voglia modificare la rappresentazione grafica del testo, modificare il foglio di stile e non anche la sua struttura semantica.
Il meccanismo dei fogli di stile si presta facilmente ad essere applicato ai documenti codificati mediante linguaggi di markup derivati da SGML e XML. Questo tipo di linguaggi, infatti, si basa proprio sulla esplicitazione degli elementi strutturali di un testo attraverso i marcatori. È sufficiente dunque definire una notazione che permetta di associare ad ogni marcatore uno stile. Naturalmente è poi necessario che il browser sia in grado di interpretare questa notazione, e di applicare le relative istruzioni di formattazione. Una notazione di questo tipo è un linguaggio per fogli di stile.
Se XML e gli standard correlati costituiscono una delle soluzioni più vantaggiose per risolvere i problemi di rappresentazione strutturale dei dati e dei documenti, i fogli di stile offrono una risposta all'esigenza di un maggiore e più raffinato controllo sulla presentazione degli stessi. L'introduzione dei fogli di stile:

- separa la specificazione della grafica dalla struttura logica del contenuto.
- consente una cura dettagliata del progetto grafico di una pagina Web.

Nell'ambito delle attività correlate allo definizione di XML, è stato creato l'*Extensible Stylesheet Language* (XSL), uno standard che permette di creare fogli di rappresentazione e trasformazione dei documenti codificati con il linguaggio XML.

A differenza delle istruzioni definite dai linguaggi di tipo procedurale, XSL ha il vantaggio di essere un documento scritto secondo la grammatica XML, e che fornisce istruzioni su come elaborare il documento al quale viene associato in sede di visualizzazione finale.
XSL si divide in due sottoinsiemi:

- *XSL Transformations* (*XSLT*, http://www.w3.org/TR/WD-xslt) è costituito da un insieme di istruzioni che permettono di applicare processi di trasformazione ai documenti XML (ad esempio dato un elenco bibliografico strutturato, una procedura XSLT può generare diversi ordinamenti) e di traduzione da una codifica XML ad un'altra o da una codifica XML a una HTML;
- *XSL Formatting Object XSL-FO*, un'applicazione XSL che fornisce una sorta di vocabolario di regole per la definizione della veste grafica. XSL FO si inserisce all'interno della grammatica XSL e fornisce le istruzioni necessarie per la formattazione, l'impaginazione e la rappresentazione del testo. Così, ad esempio, sono riproducibili
  - la differenziazione delle pagine pari e dispari
  - il layout di pagina
  - le interruzioni
  - le indentazioni
  - la grandezza e il tipo di carattere, ecc

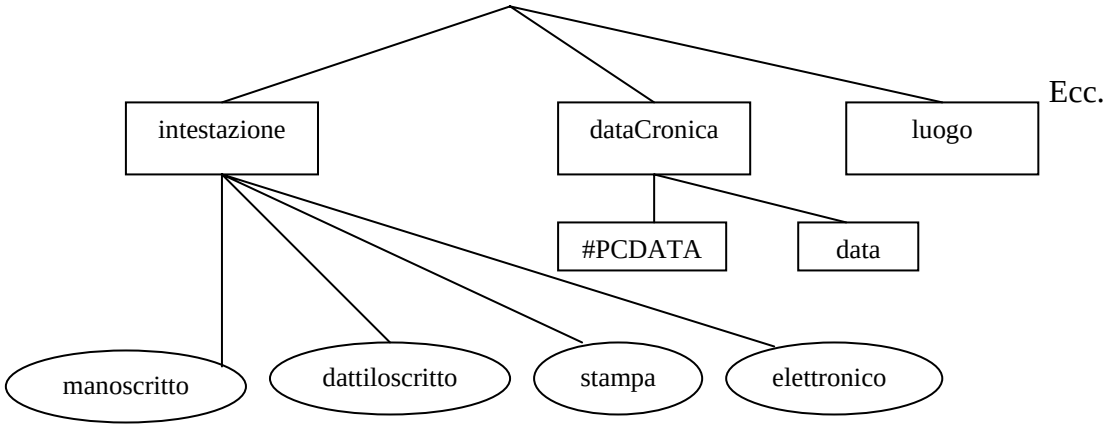
Attualmente i browser web non sono ancora in grado di supportare le istruzioni XSL-FO; in attesa che ciò si verifichi, è necessaria un'ulteriore operazione in cui l'output del documento marcato sia elaborato da un applicativo sotto forma di un formato condivisibile, come .pdf.
Extensible Stylesheet Language (http://www.w3.org/TR/WD-xsl) definisce un insieme di istruzioni di formattazione, che possono essere utilizzate per la formattazione vera e propria dei documenti e dei loro componenti. XSL permette di specificare procedure di formattazione secondo il gusto e la necessità contingenti (ad esempio: il primo paragrafo dopo un titolo non ha rientro della prima linea di testo), ed ha una flessibilità elevatissima.
Come abbiamo visto nelle pagine precedenti, un documento XML well-formed è formato da un nodo, detto radice (nel nostro caso la *Processing Instruction* presente al principio di ogni documento), il quale può contenere direttamente un solo elemento, l'elemento *root*. Questo, a sua volta, potrà comprendere sia stringhe di testo, sia altri elementi posizionati in modo da essere completamente inclusi nell'elemento in cui sono annidati.
Una struttura di questo tipo può essere rappresentata sotto forma di albero, in cui ogni elemento è rappresentato e definito da un **nodo**. Il genitore di un nodo è l'elemento in cui il nodo è annidato e i figli sono gli elementi in esso contenuti. Anche il contenuto degli elementi sono nodi, dunque anche attributi o commenti possono essere considerati come tali. In particolare possono esserci sette tipologie di nodi:

1. la radice (root node)
2. gli elementi (element node)
3. il testo (text node)
4. gli attributi (attribute node)
5. i namespaces (namespace node)
6. le processing instructions (processing instruction node)
7. i commenti (comment node)

Una rappresentazione schematica può aiutare a comprendere tali affermazioni:

inserisci esempio da documento marcato!!!!!!!!!!

diventa



Ecc.

Questa struttura ad albero è molto pratica, poiché facilita sia l'analisi (*parsing*) di un documento XML sia la conseguente elaborazione del contenuto. Essa infatti permette di accedere facilmente ai vari componenti dell'albero in base ai più svariati criteri di ricerca (vedi Allegato 4).
Chi crea un foglio di stile XSL per la trasformazione e generazione del documento XML associa all'istanza il file **.xsl** mediante una *Processing Instruction*: essa indica che il documento deve essere trasformato secondo le indicazioni contenute nel file richiamato.
Ogni nodo di questa struttura può infatti essere identificato e referenziato in maniera univoca grazie ad un linguaggio, **XPath** (path = percorso), che permette non solo di individuare un elemento specifico sulla base delle sue caratteristiche, ma anche in funzione della sua posizione all'interno dell'albero degli elementi. Non occorre che sia presente la DTD, nel senso che per utilizzarlo non ha bisogno della DTD; tuttavia trasforma una struttura da essa definita, generandola un'altra.
Ne deriva che XSL trasforma il documento XML (*source tree*, albero di partenza) in un altro file (*result tree*, nuovo albero), i cui elementi possono essere anche molto differenti da quelli iniziali o anche non esserci tutti; ciò dipende dalle scelte effettuate dall'operatore, e dalla funzione che vuole dare al documento XML trasformato.
Dovendo lavorare sulla struttura ad albero sopra illustrata, bisogna avere la possibilità di descrivere la struttura generica dell'albero, utilizzando proprio XPath: esso considera i nodi come padre, figlio, antenato.
Vi sono relazioni ben definite fra i vari nodi:

- Una radice non ha genitore, e ha come figli l'elemento che rappresenta il documento Xml, eventuali istruzioni e i commenti presenti nel prologo o dopo l'elemento «documento»;
- Un elemento ha un solo genitore e può avere zero o più figli. Questi possono essere altri elementi, commenti, istruzioni e testi, ma non attributi
- Gli attributi hanno come genitore l'elemento che li contiene, ma non ne sono figli. Può apparire un controsenso, ma è solo una questione di convenzioni. Ovviamente gli attributi non hanno figli.

Ecco alcune espressioni verbali e simboliche utilizzate nella sintassi XPath con cui sono indicati i nodi:

cosa si vuole indicare	**espressione completa**	**espressione abbreviata**	**esempio**
La root	root	/	< ?... ?>
Il singolo nodo e solo sé stesso	self :: node( )	.	sé stesso

I genitori del nodo	parent :: node( )	..	padre
Tutti i figli del nodo	children of	/*	
Il nome del figlio di quel padre	parent/child :: *nomedelfiglio*	*nomedelfiglio*	intestazione (rif. a lettera)
Attributo di un elemento	attribute :: nomedelfiglio	@ *nomedelfiglio*	@ manoscritto
Sé stesso e i discendenti della root	/descendant-or-self :: node( )/	//	lettera/dataCronica/data

La prima parte del foglio di stile richiama tutto il documento utilizzando / quale simbolo. Successivamente, per gli elementi indicati per la trasformazione da XSL, / viene utilizzato per inserire i nodi successivi della gerarchia, quasi come le cartelle e i file in esse contenute nelle directory.
Una trasformazione XSL è composta da una più regole (templates rules); ogni regola a sua volta è composta da un modello (pattern) e da un **template**, ossia una griglia, uno schema, qualunque cosa già predisposto da compilare, un modello formale, un modulo vuoto. Una volta capito dove si trova il template, esso si applica a tutti gli elementi figli o discendenti: in altre parole è ereditario.
Per fare tutto ciò il foglio di stile XSL adotta XML, è un documento XML, ossia utilizza la grammatica propria di XML per la successione e l'ordine dei tags.
Poiché in questa sede viene preso in considerazione solamente XSL come foglio di stile, il documento XSL presenta un elemento root, **xsl:stylesheet**, che presenta due attributi, *version* e *xmlns:xsl*. Tutti gli elementi che seguono sono istruzioni XSL e appartengono al namespace **xsl**, cosicché i nomi di tutti gli elementi XSL cominciano con "**xsl:**":

```
<?xml version="1.0"?>
<xsl:stylesheet    version="1.0"    xmlns:xsl="http://www.w3c.org/1999/…..vedi
meglio!!!!!!!!!!!!>
```

Cosa succede al documento XML?
Un foglio di stile può valere per più documenti XML: non c'è dunque bisogno di richiamare il documento oggetto della trasformazione XSL nel file XSL, quindi è il documento XML a richiamare in maniera univoca il foglio di stile di riferimento:
Con l'avvento di Explorer 6.0 il documento XML a cui viene associato un foglio di stile XSL sono trasformati dal Web server del Client.
Sono stati in tal modo risolti alcuni problemi che di fatto impedivano la visualizzazione sul browser dei documenti XML, e per ovviare ai quali ci si serviva di un processore interno ad un programma, nel nostro caso XML Spy, per trasformare il documento XML in un altro formato, solitamente HTML, rendendolo così gestibile dal browser.
Esiste anche un'altra possibilità per visualizzare il documento, quella per cui una macchina Server associa un foglio di stile ad uno o più documenti XML per trasformarli in un altro formato (solitamente HTML) e inviarli al Client tramite il browser. La differenza con la trasformazione operata nella soluzione precedente è che il programma invia sia al Server che al Client un documento già trasformato.

Tenendo presente l'esempio dell'Allegato 5:
- ogni template rule è un elemento **`xsl:template`**
- il pattern del rule è il valore dell'attributo **`match`** dell'elemento `xsl:template`. Esso introduce l'elemento al quale il template sarà applicato
- l'output del template è tutto ciò che viene contenuto dall'elemento `xsl:template`.

Una prima regola di trasformazione comprende:

```
<xsl:template match=”/“>
  </xsl:template>

<!-- Significa: applica il template degli elementi scritti di seguito il cui nome corrisponde a quello scritto all’interno delle virgolette. In questo caso si richiama tutto il documento, come già accennato sopra-->
```

Esistono, al di sotto, dei “sottotemplate” che specificano cosa debba essere applicato per la trasformazione indicata negli elementi del primo template, tra le virgolette. Per fare ciò un’altra istruzione XSL, **`xsl:apply-templates`**, non fa altro che dire a XSL di proseguire la trasformazione all’interno del documento che sta convertendo, nel caso in cui ci sia qualche altra regola da applicare.

```
<xsl:template match=”/“>
  <xsl:apply-templates/>
</xsl:template>

<!-- Significa: applica ricorsivamente ai template che seguiranno, se ci sono -->
```

Nel momento in cui si richiamano i nodi del documento adottando le istruzioni XSL, è poi possibile decidere quale tipo di rappresentazione si vuole. Se, ad esempio, vogliamo dare una veste grafica particolare alla lettera dell’Allegato 1, ci serviamo di tag HTML per la definizione dei caratteri, per la formattazione (vedi Allegato 5).
L’istruzione `xsl:apply-templates` può essere seguita da un attributo, **select**, il cui valore richiama un particolare figlio. Nel caso in cui tale attributo non sia presente, come nell’esempio precedente, verranno selezionati tutti i nodi figli del nodo individuato nel match del template nel quale si trova.
Un’altra istruzione frequentemente utilizzata per la creazione del foglio di stile è **`xsl:value-of`**, il quale estrae il valora di qualcosa che poi inserisce nell’output del documento finale. Tale valore viene indicato come valore dell’attributo select, in questo caso obbligatorio.

Nota esplicativa

- La differenza tra gli attributi **match** e **select** è che il primo esplicita il nodo al quale devono essere applicate le istruzioni che seguono annidate nel template, il secondo si sofferma, e richiama il contenuto del nodo. Le indicazioni di foglio di stile che effettivamente vengono visualizzate precedono il “select”, il quale applica quanto detto sopra: è una sorta di comando/conferma.
- **`xsl:apply-templates`** rinvia a dopo (“fai quello che ti dico dopo”): individua il posto in cui si vuole che vengano applicate le trasformazioni volute.
- **xsl:value-of select** indica cosa estrarre dagli elementi sopra individuati.
- Tutto quanto sia indicato come valore degli attributi delle istruzioni XSL utilizzano le espressioni XPath e possono essere indicate per esteso o in forma abbreviata. Ad esempio, per estrarre il valore dell’attributo devo far precedere il nome dell’attributo stesso dal simbolo **“@”** (vedi Allegato 5)

Nota metodologica

- Più si struttura gerarchicamente il documento in sede di costruzione della dtd, più facile è gestire il foglio di stile che gli venga applicato.
- Regole di trasformazione: sono espresse mediante template che si applica al documento: in ultima analisi si deve definire un template per ogni elemento che si vuole trasformare, anche se, in ultima istanza, si ha un template per ogni elemento.

Allegato 1: Esempio di DTD di una lettera comune

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % acc SYSTEM "accentate.txt">
%accent;
<!ENTITY dichiarazione.privacy "Si autorizza il trattamento dei dati personali ai sensi della legge 675/96">
<!ELEMENT lettera (intestazione?, dataCronica?, luogo?, salutoIniziale, corpo, salutoFinale, firma, ps?)>
<!ATTLIST lettera
    tipo-lettera (manoscritto | dattiloscritto | stampa | elettronico) #REQUIRED
>
<!ELEMENT intestazione (#PCDATA | antroponimo | qualifica | indirizzo)*>
<!ATTLIST intestazione
    tipo-intestazione (manoscritto | dattiloscritto | stampa | elettronico) #IMPLIED
>
<!ELEMENT antroponimo (pseudonimo?, (nome | cognome))+>
<!ATTLIST antroponimo
    ruolo (mittente | destinatario | altro) #REQUIRED
>
<!ELEMENT pseudonimo (#PCDATA)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT qualifica (#PCDATA)>
<!ATTLIST qualifica
    tipo-qualifica (ruolo | carica | titolo) #IMPLIED
>
<!ELEMENT indirizzo (via | civico | cap | luogo | email)*>
<!ELEMENT via (#PCDATA)>
<!ELEMENT civico (#PCDATA)>
<!ELEMENT cap (#PCDATA)>
<!ELEMENT luogo (#PCDATA)>
<!ATTLIST luogo
    tipo-luogo (citta | provincia | localita | frazione | comune | stato) #IMPLIED
>
<!ELEMENT email (#PCDATA)>
<!ELEMENT dataCronica (#PCDATA | data)*>
<!ATTLIST dataCronica
    datacronicaNorm CDATA #IMPLIED
>
<!ELEMENT data (giorno | mese | anno)*>
<!ELEMENT giorno (#PCDATA)>
<!ELEMENT mese (#PCDATA)>
<!ELEMENT anno (#PCDATA)>
<!ELEMENT salutoIniziale (#PCDATA | appellativo | qualifica | antroponimo)*>
<!ELEMENT appellativo (#PCDATA)>
<!ELEMENT corpo (#PCDATA | antroponimo | luogo | ente | data)*>
<!ELEMENT ente (#PCDATA)>
<!ELEMENT salutoFinale (#PCDATA | saluto)*>
<!ELEMENT saluto (#PCDATA)>
<!ELEMENT firma (#PCDATA | antroponimo)*>
<!ATTLIST firma
    tipofirma (manoscritta | dattiloscritta | timbro) #IMPLIED
>
<!ELEMENT ps (corpo | firma)>
```

ALLEGATO 2: Accentate.txt – File richiamato nella entità esterna parametrica

```
<!ENTITY agrave "&#225;">
<!-- á -->
<!ENTITY aacute "&#224;">
<!-- à -->
<!ENTITY egrave "&#232;">
<!-- è -->
<!ENTITY eacute "&#233;">
<!-- é -->
<!ENTITY igrave "&#236;">
<!-- ì -->
<!ENTITY iacute "&#237;">
<!-- í -->
<!ENTITY ograve "&#242;">
<!-- ò -->
<!ENTITY oacute "&#243;">
<!-- ó -->
<!ENTITY ugrave "&#249;">
<!-- ù -->
<!ENTITY uacute "&#250;">
<!-- ú -->
<!ENTITY Egrave "&#200;">
<!-- È -->
<!ENTITY icirc "&#238;">
<!-- î -->
```

ALLEGATO 3: Testo di una lettera

Caro nuovo amico,

consenti non solo al più vecchio del nostro club, ma – credo – a uno dei più vecchi e anziani dell'intero distretto, di esprimerti il proprio pensiero e anche qualche consiglio.

E' passato quasi mezzo secolo da quando, con alcuni amici, ho fondato, nel 1948, il nostro club e da allora ho assistito alla profonda evoluzione del nostro Rotary come, forse, doveva avvenire.

Molte cose sono cambiate: il Rotary si è sempre più diffuso nel mondo, i distretti, i clubs e i soci sono enormemente aumentati. Ma sono enormemente cresciute le responsabilità; non tanto negli ideali, ovviamente immutati, quanto nella difficoltà di estendere i rapporti personali, tanto importanti per mantenere lo spirito rotaryano.

Consentimi un'osservazione del tutto personale e certamente non condivisa in alto loco: la proliferazione dei distretti e dei clubs ha ristretto la possibilità dei contatti e dei rapporti di amicizia fra i rotaryani.

Ricordo con nostalgia le Assemblee, quando il distretto era unico, le opportunità di incontri coi rotaryani dei clubs anche più lontani e i nascenti rapporti di conoscenza e di amicizia.

Tutto cambia, e pure il Rotary, per adattarsi all'inevitabile evoluzione del mondo e della società.

Ma questo cambiamento non deve e non può incidere sulla vita e sugli ideali del Rotary, quali furono fissati da Paul Harrys. Non posso quindi che richiamare alla tua attenzione certi principi fondamentali: il sentimento di amicizia, base di ogni associazionismo, la lealtà verso tutti, l'esempio per la famiglia e per la società e, è superfluo dirlo, l'onestà.

Per statuto sei tenuto a relazioni su argomenti interessanti. Non dimenticare però che ti rivolgi a persone di diversa cultura e di diversi interessi. Pur se curate, le relazioni siano di facile comprensione e limitate nel tempo. Diversamente che all'estero, dove gli interventi sono ridotti per la brevità dei meetings, da noi sono allietati da cibi abbondanti che favoriscono le distrazioni. Fa che i rotaryani non sentano con sollievo il suono della campana.

E quando sarà richiesto il tuo voto, dallo con coscienza delle doti rotaryane e col solo riguardo alle qualità richieste. Evita personalismi e legami di altra natura.

Ricorda, soprattutto, che sei divenuto rotaryano.
Un caro saluto. Il vecchio, anzi vecchissimo

Massimo Stanghellini Perilli

ALLEGATO 4: La lettera dell'Allegato 3 codifica in XML e validata dalla DTD riportata nell'Allegato 1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE lettera SYSTEM "C:\WINDOWS\Desktop\ilaria\Napoli-SVB\PerfezNapoli\materialeXML-
EAD\matDef\esempiXML\lettera.dtd">
<?xml-stylesheet type="text/xsl" href="C:\WINDOWS\Desktop\ilaria\Napoli-SVB\PerfezNapoli\materialeXML-
EAD\matDef\esempiXML\lettera.xsl"?>
<lettera tipo-lettera="elettronico">
    <salutoIniziale> Caro nuovo amico,
</salutoIniziale>
    <corpo>
    consenti non solo al pi&ugrave; vecchio del nostro club, ma – credo – a uno dei più vecchi e anziani dell'intero distretto, di esprimerti il proprio pensiero e anche qualche consiglio.&Egrave; passato quasi mezzo secolo da quando, con alcuni amici, ho fondato, nel 1948, il nostro club e da allora ho assistito alla profonda evoluzione del nostro Rotary come, forse, doveva avvenire.

Molte cose sono cambiate: il Rotary si &egrave; sempre più diffuso nel mondo, i distretti, i clubs e i soci sono enormemente aumentati. Ma sono enormemente cresciute le responsabilit&aacute;; non tanto negli ideali, ovviamente immutati, quanto nella difficolt&aacute; di estendere i rapporti personali, tanto importanti per mantenere lo spirito rotaryano.

Consentimi un'osservazione del tutto personale e certamente non condivisa in alto loco: la proliferazione dei distretti e dei clubs ha ristretto la possibilit&aacute; dei contatti e dei rapporti di amicizia fra i rotaryani.

Ricordo con nostalgia le Assemblee, quando il distretto era unico, le opportunit&aacute; di incontri coi rotaryani dei clubs anche più lontani e i nascenti rapporti di conoscenza e di amicizia.

Tutto cambia, e pure il Rotary, per adattarsi all'inevitabile evoluzione del mondo e della societ&aacute;.

Ma questo cambiamento non deve e non può incidere sulla vita e sugli ideali del Rotary, quali furono fissati da Paul Harrys. Non posso quindi che richiamare alla tua attenzione certi principi fondamentali: il sentimento di amicizia, base di ogni associazionismo, la lealt&aacute; verso tutti, l'esempio per la famiglia e per la societ&aacute; e, &egrave; superfluo dirlo, l'onest&agrave;.

Per statuto sei tenuto a relazioni su argomenti interessanti. Non dimenticare però che ti rivolgi a persone di diversa cultura e di diversi interessi. Pur se curate, le relazioni siano di facile comprensione e limitate nel tempo. Diversamente che all'estero, dove gli interventi sono ridotti per la brevit&aacute; dei meetings, da noi sono allietati da cibi abbondanti che favoriscono le distrazioni. Fa che i rotaryani non sentano con sollievo il suono della campana.

E quando sar&aacute; richiesto il tuo voto, dallo con coscienza delle doti rotaryane e col solo riguardo alle qualit&aacute; richieste. Evita personalismi e legami di altra natura.

Ricorda, soprattutto, che sei divenuto rotaryano.
</corpo>
    <salutoFinale>Un caro saluto.</salutoFinale>
    <firma tipofirma="dattiloscritta">Il vecchio, anzi vecchissimo
<antroponimo ruolo="mittente">
            <nome>Massimo Stanghellini Perilli &dichiarazione.privacy;</nome>
            <!--nota la entity-->
        </antroponimo>
    </firma>
</lettera>
```