Guida all'XML

(corso online a cura di Marco Giannì; http://www.html.it/xml/guida/)

INTRODUZIONE

L'HTML (Hypertext Markup Language) è considerato la base del World Wide Web. Questo linguaggio consente infatti di creare in maniera standardizzata pagine di informazioni formattate in grado di raggiungere, tramite Internet, un numero di utenti in costante aumento. Insieme al protocollo HTTP (Hypertext Transport Protocol), l'HTML ha rivoluzionato il modo in cui le persone inviano e ricevono informazioni, ma lo scopo principale per cui è stato realizzato è la *visualizzazione* dei dati. Per questo motivo, l'HTML prende in considerazione soprattutto il modo in cui le informazioni vengono presentate e non il tipo o la struttura di tali informazioni, aspetti per i quali è stato sviluppato il linguaggio XML (eXtensible Markup Language). La necessità di espandere le capacità di HTML ha spinto i produttori di browser a introdurre nuovi marcatori nella sintassi, rendendola a tutti gli effetti proprietaria e non più standard. Da ciò segue che una pagina HTML che sfrutti marcatori proprietari non può essere visualizzata correttamente se non con il browser adatto, con le ovvie conseguenze che ne derivano.
L'XML è un linguaggio di markup aperto e basato su testo che fornisce informazioni di tipo strutturale e semantico relative ai dati veri e propri. Questi "dati sui dati", o *metadati*, offrono un contesto aggiuntivo all'applicazione che utilizza i dati e consente un nuovo livello di gestione e manipolazione delle informazioni basate su Web.
L'XML, derivazione del noto linguaggio SGML (Standard Generalized Markup Language), è stato ottimizzato per il Web, diventando potente complemento dell'HTML basato su standard. L'importanza dell'XML nel futuro delle informazioni sul Web potrebbe pertanto giungere ad eguagliare quella dell'HTML agli albori.

CAPITOLO 1. L'IMPORTANZA DI XML

1.1 ORIGINE E OBIETTIVI

L'Extensible Markup Language (XML) è un metalinguaggio che permette di creare dei linguaggi personalizzati di markup; nasce dall'esigenza di portare nel World Wide Web lo Standard Generalized Markup Language (SGML), lo standard internazionale per la descrizione della struttura e del contenuto di documenti elettronici di qualsiasi tipo; ne contiene quindi tutta la potenza, ma non tutte le complesse funzioni raramente utilizzate. Si caratterizza per la semplicità con cui è possibile scrivere documenti, condividerli e trasmetterli nel Web.
L'utilizzo di XML permette di superare il grosso limite attuale del Web, che è quello della dipendenza da un tipo di documento HTML, singolo e non estensibile. Questo linguaggio è nato per permettere agli utenti del World Wide Web di condividere le informazioni su sistemi differenti; il presupposto era che quelle informazioni fossero testo con al più alcune immagini e collegamenti ipertestuali. Attualmente però, le informazioni sul World Wide Web sono database di testo, immagini, suoni, video, audio. Quindi l'HTML è stato chiamato sempre più di frequente a fornire soluzioni a problemi che non aveva lo scopo di risolvere, come dover descrivere tipi differenti e specifici di informazioni, definire relazioni complesse di collegamenti fra documenti, trasmettere informazioni in diversi formati. Per

superare questi problemi, sono state create delle estensioni dell'HTML, spesso fra loro incompatibili.
L'XML permette a gruppi di persone o ad organizzazioni di creare il proprio linguaggio di markup, specifico per il tipo di informazione che trattano; per molte applicazioni e per diversi settori, gli esperti hanno già creato linguaggi di markup specifici, come ad esempio il Channel Definition Format, il Mathematical Markup Language ed altri (Fig. 1.1).

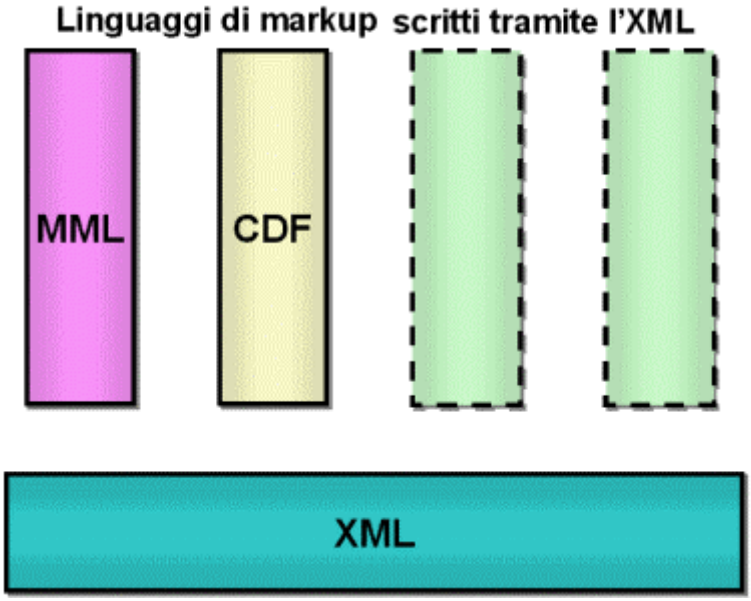


Fig. 1.1: L'XML permette di scrivere linguaggi di markup

XML fu sviluppato da XML Working Group (originariamente noto come SGML Editorial Review Board) costituitosi sotto gli auspici del World Wide Web Consortium (W3C) nel 1996. Esso era presieduto da Jon Bosak della Sun Microsystems con la partecipazione attiva dell'XML Special Interest Group (precedentemente noto come SGML Working Group) anch'esso organizzato dal W3C.
L'obiettivo di questo gruppo di lavoro era di portare il linguaggio SGML nel Web. L'SGML è un linguaggio per la specifica dei linguaggi di markup ed è il genitore del ben noto HTML.
La progettazione dell'XML venne eseguita esaminando i punti di forza e di debolezza dell'SGML. Il risultato è uno standard per i linguaggi di markup che contiene tutta la potenza dell'SGML ma non tutte le funzioni complesse e raramente utilizzate. L'XML venne mostrato per la prima volta al pubblico quando l'SGML celebrò il suo decimo anno.
Gli obiettivi progettuali di XML sono:

1. **XML deve essere utilizzabile in modo semplice su Internet**: in primo luogo, l'XML deve operare in maniera efficiente su Internet e soddisfare le esigenze delle applicazioni eseguite in un ambiente di rete distribuito.

2. **XML deve supportare un gran numero di applicazioni**: deve essere possibile utilizzare l'XML con un'ampia gamma di applicazioni, tra cui strumenti di creazione, motori per la visualizzazione di contenuti, strumenti di traduzione e applicazioni di database.

3. **XML deve essere compatibile con SGML**: questo obiettivo è stato definito sulla base del presupposto che un documento XML valido debba anche essere un documento SGML valido, in modo tale che gli strumenti SGML esistenti possano essere utilizzati con l'XML e siano in grado di *analizzare* il codice XML.

4. **Deve essere facile lo sviluppo di programmi che elaborino documenti XML**: l'adozione del linguaggio è proporzionale alla disponibilità di strumenti e la proliferazione di questi è la dimostrazione che questo obiettivo è stato raggiunto.

5. **Il numero di caratteristiche opzionali deve essere mantenuto al minimo possibile**: al contrario dell'SGML, l'XML elimina le opzioni, in tal modo qualsiasi elaboratore potrà pertanto analizzare qualunque documento XML, indipendentemente dai dati e dalla struttura contenuti nel documento.

6. **I documenti XML dovrebbero essere leggibili da un utente e ragionevolmente chiari**: poiché utilizza il testo normale per descrivere i dati e le relazioni tra i dati, l'XML è più semplice da utilizzare e da leggere del formato binario che esegue la stessa operazione; inoltre poiché il codice è formattato in modo diretto, è utile che l'XML sia facilmente leggibile da parte sia degli utenti che dei computer.

7. **La progettazione di XML dovrebbe essere rapida**: l'XML è stato sviluppato per soddisfare l'esigenza di un linguaggio estensibile per il Web. Questo obiettivo è stato definito dopo aver considerato l'eventualità che se l'XML non fosse stato reso disponibile rapidamente come metodo per estendere l'HTML, altre organizzazioni avrebbero potuto provvedere a fornire una soluzione proprietaria, binaria o entrambe.

8. **La progettazione di XML deve essere formale e concisa**: questo obiettivo deriva dall'esigenza di rendere il linguaggio il più possibile conciso, formalizzando la formulazione della specifica.

9. **I documenti XML devono essere facili da creare**: i documenti XML possono essere creati facendo ricorso a strumenti di semplice utilizzo, quali editor di testo normale.

10. **Non è di nessuna importanza l'economicità nel markup XML**: nell'SGML e nell'HTML la presenza di un tag di apertura è sufficiente per segnalare che l'elemento precedente deve essere chiuso. Benché così sia possibile ridurre il lavoro degli autori, questa soluzione potrebbe essere fonte di confusione per i lettori, nell'XML la chiarezza ha in ogni caso la precedenza sulla concisione.

1.2 RELAZIONI TRA XML E ALTRI LINGUAGGI

1.2.1 RELAZIONE TRA XML E SGML

SGML è un meta-linguaggio, ovvero un insieme di regole generalizzate usate per creare molteplici linguaggi speciali che prendono il nome di markup language. Le applicazioni più note di SGML sono HTML e TIM (Telecommunication Interchange Markup); inoltre SGML viene largamente usato dalle grandi industrie di tecnologia come strumento di immagazzinamento e scambio di informazioni di qualsiasi tipo. E' poi utilizzato in tipografia (ad esempio nella stampa delle delibere) per creare il documento che poi le macchine tipografiche andranno a stampare secondo la loro interpretazione specifica delle strutture SGML.

L’SGML stesso non ha mai suscitato particolare interesse tra gli sviluppatori Web, probabilmente a causa soprattutto della sua complessità e delle difficoltà che l’utilizzo di questo linguaggio comporta.
L’obiettivo era di includere nell’XML, solo le parti dell’SGML necessarie per la pubblicazione sul Web. XML eredita da SGML la capacità di definire con estrema facilità nuovi marcatori, creando di fatto dei linguaggi di markup personalizzati, mentre la complessità e le caratteristiche opzionali che appesantivano l’SGML sono state pertanto eliminate dall’XML.

Pregi di SGML :
La sua potenza è la flessibilità. E' uno standard (ISO 8879:1986) che può essere applicato ad ogni tipo dato; è espandibile ed è fortemente strutturato. Queste caratteristiche lo rendono capace di risolvere problemi di elaborazione dell'informazione tra i più complessi, garantendo un perfetto riutilizzo dei dati. Inoltre è non-proprietario ed indipendente dalla piattaforma.

Limiti di SGML :
Ha una struttura molto pesante, comprensiva di un gran numero di opzioni che lo rendono poco maneggevole da un eventuale software di manipolazione dei dati, quindi può rappresentare un grosso scoglio per i programmatori che lavorano sul Web. Comunque sia il problema più importante è che SGML richiede un DTD (Document Type Definition) per l'identificazione di tutte le relazioni tra le entità che compongono il documento SGML, e di uno Style-Sheet cioè un foglio di stile che stabilisce il modo di rappresentare e/o visualizzare i dati e le strutture definite nel DTD. Inoltre ogni documento SGML deve necessariamente essere validato e quindi oltre ad un controllo sintattico viene controllata anche la semantica delle strutture di dati presenti nel documento (ad esempio non potrà succedere che un titolo di capitolo viene dopo il relativo sottotitolo). Si capisce quindi che in mancanza di uno di questi elementi, o in caso di corruzione dei dati, le istanze SGML saranno visualizzate come codice incomprensibile per l'utente finale. Il risultato di queste considerazioni è che per le applicazioni sul World Wide Web (WWW) le istanze SGML non sono praticamente portabili data la loro intrinseca pesantezza e non robustezza.

1.2.2 RELAZIONE TRA XML E HTML

L’XML è spesso considerato una sostituzione dell’HTML. Sebbene questo possa essere in parte vero, in realtà i due linguaggi sono complementari e, relativamente al modo in cui vengono trattati i dati, operano su livelli differenti. Nei casi in cui l’XML viene utilizzato per strutturare e descrivere i dati sul Web, l’HTML è usato per formattare i dati.

Pregi di HTML :
Data la semplicità della struttura dei documenti HTML, rappresenta un veloce strumento per lo scambio delle informazioni sul Web. Le istanze HTML non devono essere validate ma è sufficiente che siano "ben formate", ovvero sintatticamente corrette. Il modo con cui viene visualizzato il documento dipende dallo Style-Sheet del browser stesso.

Limiti di HTML :
I limiti di HTML discendono proprio dai suoi pregi e sono dovuti soprattutto ad un uso distorto di questo strumento. Dal momento che non è richiesta una strutturazione semanticamente corretta delle informazioni, HTML diventa molto povero per le applicazioni di elaborazione dei dati, quindi il problema viene completamente demandato

agli applicativi scritti in Java, Java Script, ecc. La sua struttura non è estensibile e questo lo rende praticamente inutilizzabile per le industrie che si sono viste costrette a crearsi standards differenti per ogni diversa applicazione, ed ancora una volta è compito del software elaborare i dati per il trasferimento delle informazioni sul Web. Un altro problema da non sottovalutare si può verificare dalla parte del client qualora non abbia un dispositivo video del tipo previsto dal server (per esempio dispositivi per il braille o monitor non grafici).

Inoltre l'HTML non offre i meccanismi per mantenere il controllo della formattazione. Infatti non si possono specificare le dimensioni video di un documento o controllare le dimensioni della finestra di un browser. Per superare questo problema sono stati inseriti nuovi tag e i fogli di stile; i nuovi tag forniscono istruzioni di formattazione come ad esempio il tag <FONT> per specificare il tipo di carattere; gli utenti però, possono ignorare queste specifiche e utilizzare le loro; inoltre questi tag sono di formattazione e non di descrizione (come dovrebbero essere essendo l'HTML un linguaggio di descrizione del documento). Tutto questo fa sì che uno sviluppatore di pagine Web non sappia mai con certezza cosa un client visualizzerà sullo schermo del computer.
Il W3C (ente guida per lo sviluppo del Web), si rese conto che la creazione di una moltitudine di nuovi tag che rispondessero a ogni possibile esigenza di formattazione era irrealistica ed incoerente con i principi ed i concetti dell'HTML; vennero quindi introdotti i CSS (Cascading Style Sheet) che permettono attraverso le norme di stile di definire come devono apparire determinati elementi di un documento; anche con i CSS però, si deve scegliere fra un assortimento di proprietà predefinite.
Bisogna anche specificare che lasciare al browser la decisione di come presentare un documento, fu esplicitamente voluto come impostazione del progetto HTML (infatti come si è già detto è un linguaggio di descrizione del documento). Il browser conosce le preferenze degli utenti e soprattutto l'ambiente in cui si trova ad operare; tutte informazioni che logicamente l'autore di un documento HTML non conosce (a parte pochi casi, come un ambiente intranet omogeneo). Il poter decidere come visualizzare un documento è molto utile ad esempio quando un utente non dispone di un browser grafico, oppure per utenti che hanno problemi di vista e hanno bisogno di caratteri molto grandi, oppure addirittura per utenti non vedenti. Sfortunatamente, le case produttrici di browser, non hanno capito la filosofia di base dell'HTML; il risultato di questo è che allo stato attuale un numero enorme di pagine sul Web, contengono insiemi di tag specifici per determinati browser, e anche per determinate versioni dello stesso browser; queste pagine sono molto spesso illeggibili. Gradualmente l'HTML sta passando da un linguaggio di descrizione, ad un linguaggio di presentazione, con tutti i problemi qui descritti; l'XML può fornire una soluzione, fornendo un layout a chiunque e indipendentemente dal browser utilizzato, attraverso l'utilizzo dell'Exstensible Style Language (XSL), che diventerà uno standard universalmente accettato.

L'XML si differenzia dall'HTML per tre maggiori aspetti:

- Possono essere definiti nuovi tag ed attributi.

- La struttura di un documento può essere vista in modo gerarchico nidificando i tag in ogni livello di complessità.

- Ogni documento XML può contenere una opzionale descrizione della sua grammatica, in modo che possa essere utilizzata da applicazioni che richiedono una validazione della struttura del documento.

1.3 LE PROPRIETÀ DELL'XML

L'XML è importante in due classi di applicazioni Web: la creazione di documenti e lo scambio dei dati; i server Web attualmente utilizzati richiedono, per essere in grado di servire documenti XML, minime modifiche di configurazione; inoltre il metodo standard di collegamento e la connessione dei documenti XML, utilizza gli URL, che vengono interpretati correttamente della maggior parte del software per Internet.
La sintassi dell'XML è molto simile a quella dell'HTML, ma molto più rigida e severa; anche se al primo impatto questa caratteristica non sembra una proprietà positiva, (soprattutto per chi dovrà scrivere documenti XML), è stata volutamente introdotta dal gruppo di studio del W3C, per facilitare lo sviluppo di applicazioni basate sull'XML e aumentarne le prestazioni (si pensi ad un browser). Inoltre una sintassi chiara e pulita, aumenta la leggibilità di un documento (questo vale in generale); a questo proposito si può dire che una sintassi chiara e pulita, unita alla possibilità di creare un proprio set di markup, contribuirà a rendere un file XML leggibile quanto un file di solo testo (e in alcuni casi di più).
L'XML non è limitato a un insieme fisso di tipi di elementi, ma permette di definire e utilizzare elementi e attributi personalizzati; per far questo viene fornita una sintassi con cui è possibile specificare gli elementi e gli attributi che possono essere utilizzati all'interno dei documenti. In altre parole è possibile creare un modello, chiamato Document Type Definition (DTD), che descrive la struttura e il contenuto di una classe di documenti; lo stesso XML ha un proprio DTD (attualmente descritto nella specifica REC-xml-19980210) in cui vengono elencate le regole della specifica stessa del linguaggio. Con l'XML è anche introdotta una classe di documenti che fa riferimento al solo DTD dell'XML; la creazione di un DTD personale non è quindi indispensabile, come mostra la Fig. 1.2.

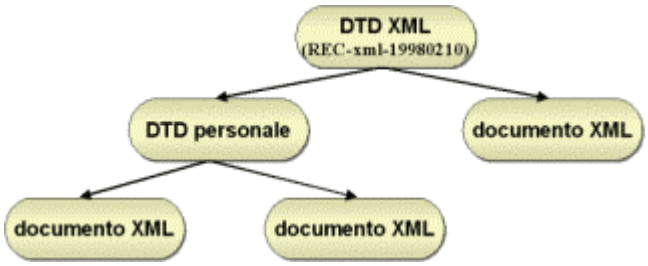


Fig. 1.2: Un documento XML può fare riferimento
al DTD dell'XML o a un DTD personale

Questa possibilità semplifica molto l'utilizzo dell'XML rispetto all'SGML. Nel linguaggio SGML infatti i DTD sono indispensabili (e spesso complicati da creare e da imparare), e un documento deve fare obbligatoriamente riferimento ad uno di essi.
L'XML permette di creare dei tag personalizzati; inoltre uno stesso documento XML può essere utilizzato per scopi diversi da applicazioni diverse. Come fa una applicazione a riconoscere il markup disegnato per lei e ad evitare di confonderlo con il markup disegnato per altre applicazioni? Ad esempio una applicazione potrebbe utilizzare un elemento chiamato "address" per identificare il domicilio di una persona; un'altra applicazione invece potrebbe utilizzare lo stesso elemento per identificare l'indirizzo elettronico di una persona. Anche un programmatore potrebbe avere difficoltà nel capire l'utilizzo di un determinato elemento; per risolvere questo tipo di problemi, il gruppo di lavoro del W3C ha pensato ad un metodo per individuare le convenzioni che governano l'utilizzo di un particolare set di

elementi; l'idea è quella di utilizzare un namespace, cioè un documento in cui viene definito l'utilizzo di un particolare set di elementi; un documento XML può far riferimento ad un namespace attraverso un indirizzo Web. Il documento di riferimento è il WD-xml-names-19990114.
L'XML può essere utilizzato come piattaforma per lo scambio di dati tra le applicazioni, come mostra la Fig. 1.3; ciò è possibile perché è orientato alla descrizione dei dati.

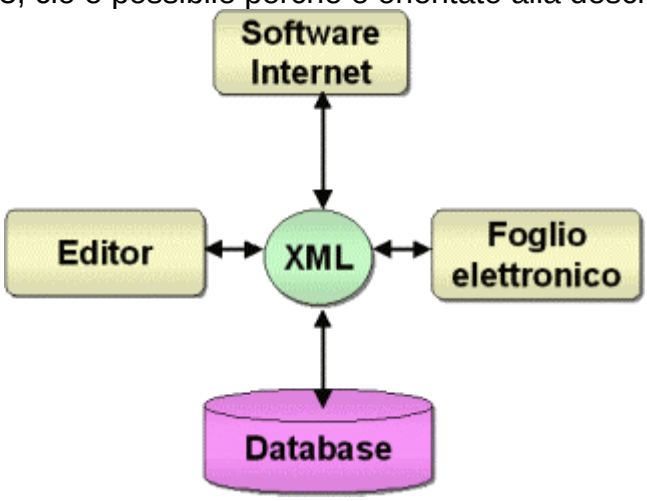


Fig 1.3: L'XML può essere utilizzato come piattaforma per lo scambio di dati

Poniamo il caso che si voglia scambiare le informazioni di database su Internet. Si immagini di utilizzare un browser per rinviare al server le informazioni su un questionario compilato dagli utenti. Questo processo, come molti altri, richiede un formato che possa essere personalizzato per un utilizzo specifico e che sia una soluzione aperta non proprietaria.
L'XML è la soluzione per questo tipo di problema. Questo linguaggio in futuro diventerà sempre più importante per lo scambio di dati su Internet.
L'XML è in grado di fornire una sola piattaforma per lo scambio di dati tra le applicazioni. Era sempre stato difficile trovare un formato di interscambio che potesse essere utilizzato per il trasferimento di dati tra database di fornitori differenti e sistemi operativi diversi. Quel tipo di interscambio è ora diventato una delle applicazioni principali dell'XML.
Le pagine HTML hanno l'unico scopo di essere visualizzate da un browser (infatti si dice che i dati nell'HTML sono orientati al video); per questo è molto difficile l'elaborazione successiva delle informazioni contenute nelle pagine HTML. I documenti basati sull'XML invece, non fanno supposizioni su come verranno utilizzati dal client; così le informazioni ricevute possono essere utilizzate da un'applicazione che comprende il linguaggio XML, utilizzando i dati ivi contenuti in altri processi software; quindi uno stesso documento può essere facilmente utilizzato per scopi diversi.
Il collegamento ipertestuale è una caratteristica specifica dell'HTML; attualmente però, esso supporta solo un tipo di collegamento, che è quello unidirezionale; in un vero sistema ipertestuale i tipi di collegamento sono diversi. Anche se l'XML è uno standard, molte cose sulle tecnologie correlate, quali i fogli di stile ed il collegamento, sono ancora in fase di sviluppo; quindi il modo esatto in cui il collegamento deve essere implementato nell'XML è ancora in fase di studio. Sicuramente dovrà essere compatibile con i meccanismi di collegamento HTML esistenti, supportare l'estensibilità e le proprietà intrinseche dell'XML e implementare i vari tipi di collegamenti propri di un vero sistema ipertestuale. Attualmente lo standard di riferimento è l'Extensible Linking Languge (XLL) del 3/3/1998; è stato diviso in due parti: XML Linking Language (XLink) e XML Pointer Language (XPointer).

L'XSL definisce la specifica per la presentazione e l'aspetto di un documento XML: è stato presentato nel 1997 da un consorzio di industrie software (tra cui anche la Microsoft) al W3C perché lo approvasse come linguaggio di stile standard per i documenti XML. L'XSL è un sottoinsieme del Document Style Semantics and Specification Language (DSSSL), il linguaggio di stile utilizzato in ambiente SGML; gode delle proprietà di essere estensibile, potente ma nello stesso tempo di facile utilizzo.
Con l'XSL è possibile creare fogli di stile che permettono la visualizzazione di un documento XML in un qualsiasi formato (audio, video, braille, etc.), come mostra la Fig. 1.4. Attualmente lo standard di riferimento è il documento WD-xsl-19990421.

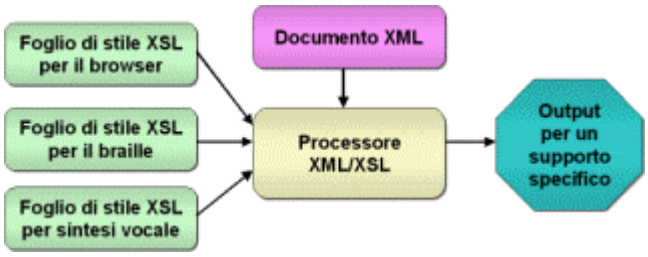


Fig 1.4: L'XSL permette di creare fogli di stile per un qualsiasi formato

1.4 L'XML E IL WEB

Si possono individuare quattro categorie di applicazioni che utilizzeranno l'XML:

- Applicazioni che richiedono al Web client di mediare tra due o più database eterogenei.
- Applicazioni che tentano di distribuire i processi di caricamento dell'informazione dal Web server al Web client.
- Applicazioni che richiedono al Web client di presentare differenti viste degli stessi dati ad utenti differenti.
- Applicazioni nelle quali motori di ricerca Web tentano di ritagliare le informazioni scoperte, ai bisogni individuali degli utenti.

L'alternativa all'XML, per queste applicazioni, sono codici proprietari inseriti come script nei documenti HTML. La filosofia dell'XML invece, si basa sul fatto che il formato dei dati non sia legato a nessuno script in particolare, e sia uno standard indipendente da qualsiasi organizzazione proprietaria.

1.4.1 DATABASE ETEROGENEI

Un esempio di questa categoria di applicazioni Web è il sistema informativo di un'agenzia americana di "home health care". Questo tipo di agenzie sono il maggior componente dell'industria medica americana e la loro importanza sta aumentando da quando politiche governative hanno spostato l'assistenza medica ospedaliera verso l'assistenza medica domestica. Come si può ben capire la gestione dell'informazione è critica per questa industria; la salute di un paziente è rappresentata nel sistema informativo attraverso una collezione di documenti storici che rappresentano la vita medica di una persona, passata attraverso vari dottori, ospedali, farmacie e compagnie di assicurazioni; quando un nuovo paziente entra in una agenzia, c'è l'enorme compito di prelevare tutto il materiale e di

memorizzarlo nel database dell'agenzia. L'avvento del Web diede alla comunità informatica medica la speranza di poter semplificare lo sforzo di memorizzazione delle informazioni nel database; sfortunatamente le applicazioni Web esistenti offrono modelli di soluzione a questo problema inadeguati. Gli ospedali offrono alle agenzie una soluzione che in poche parole è così riassunta:

1. Raggiungere il sito Web dell'ospedale
2. Diventare un utente autorizzato
3. Accedere alla documentazione medica del paziente attraverso il browser
4. Stampare la documentazione
5. Inserire manualmente i dati nel database (dalla stampa)

Attualmente questa soluzione è proposta da un gran numero di ospedali americani. Una versione leggermente più sofisticata permette all'operatore di inserire manualmente i dati letti dal browser direttamente in un form dell'agenzia (in una finestra separata), evitando così la stampa del documento. Anche questa però non è una grande soluzione. La soluzione ideale sarebbe la seguente:

1. Raggiungere il sito Web dell'ospedale
2. Diventare un utente autorizzato
3. Accedere alla documentazione medica del paziente attraverso una interfaccia Web che rappresenti la documentazione con una icona a cartella
4. Fare un drag della cartella dall'applicazione Web nel database interno
5. Fare un drop della cartella nel database

Attualmente questa soluzione non è possibile poiché ci si scontra con i limiti dell'HTML; le ragioni sono due:

- L'HTML non permette di rappresentare strutture dati
- L'HTML non permette il controllo dei dati per validare i documenti ricevuti

Una soluzione tecnica per implementare questo scambio di documenti è quella di richiedere agli ospedali e alle agenzie di utilizzare un sistema informativo standard dettato dal governo (tale soluzione è attualmente allo studio); questo tipo di soluzione è però difficile da mettere in pratica, soprattutto in un ambiente dove ospedali e agenzie stanno attraversando un momento di difficoltà finanziaria (cambiare il sistema informativo comporta generalmente grosse spese). Un'altra soluzione è quella di adottare un formato standard di scambio dell'informazione; un grande numero di industrie nel campo spaziale, telecomunicazioni, hardware, software, ha utilizzato per anni un linguaggio standard per lo scambio dei dati e il processo è attualmente molto ben compreso. Tipicamente un consorzio di grandi industrie definisce un Document Type Definition (in ambiente SGML) per implementare un linguaggio di markup specifico per un determinato scopo; quindi il linguaggio è utilizzato come standard per lo scambio dei dati in determinati ambienti.
La soluzione XML è indipendente dai sistemi, dalle organizzazioni e proviene dalla decennale esperienza dell'SGML; l'XML permette di utilizzare l'approccio SGML per lo scambio dei dati nel Web; è significativo come il giorno del rilascio della prima versione stabile dell'XML, l'organizzazione che raggruppa le maggiori agenzie di home health care, abbia annunciato lo sviluppo dell'Health Care Markup Language in ambiente SGML, che dovrebbe risolvere i tipi di problemi descritti in questo esempio.
Si è anche dimostrato che rappresentare i dati con un ricco markup ha dei benefici che vanno oltre lo scambio dei dati; ad esempio è molto utile rappresentare risultati di un

esame clinico con tag quali <allergia> oppure <reazione>; infatti chi legge il documento è subito allertato (da una applicazione apposita) del fatto che un paziente può essere allergico alla penicillina.

1.4.2 PROCESSI DISTRIBUITI

Un esempio di questa seconda categoria di applicazioni XML è il sistema di distribuzione dei dati adottato dall'industria dei semiconduttori. Ogni grande industria nel campo dei semiconduttori deve mantenere enormi quantità di dati tecnici sui circuiti integrati prodotti. Per abilitare lo scambio di questi dati, anni fa fu formato un consorzio (il Pinnacles Group) di industrie quali Intel, National Semiconductor, Philips, Texas Instrument e Hitachi; lo scopo era quello di sviluppare uno specifico linguaggio di markup in ambiente SGML; nel 1995 è stata presentata la prima versione stabile e attualmente le grandi compagnie sono impegnate nel processo d'implementazione di questo linguaggio. Si potrebbe pensare che l'incremento della popolarità dell'HTML avesse fatto cambiare idea ai membri del Pinnacles Group, ma le limitazioni di tale linguaggio li hanno convinti che l'idea originale fosse più che corretta. L'idea era che utilizzare il linguaggio di markup come veicolo per la distribuzione dei dati sui circuiti integrati potesse permettere non solo la loro visualizzazione, ma anche il progetto dei circuiti stessi. Questo approccio si integra molto bene con la tecnologia dei Java applet perché permette ad un ingegnere di accedere al sito Web di una industria di semiconduttori e di scaricarsi non solo i dati di un particolare circuito integrato, ma anche un Java applet che permetta di combinare i dati in vari modi.
Questo esempio dei semiconduttori è una buona dimostrazione dei vantaggi dell'XML perché:

- Richiede uno specifico tag set che non può essere ottenuto con il non estensibile tag set dell'HTML.
- Richiede che la rappresentazione dei dati sia indipendente dai sistemi utilizzati nelle varie industrie.

Creare il disegno dei circuiti dai dati è un processo computazionalmente molto intensivo; quindi in un ambiente Web client-server è necessario distribuire il processo computazionale per ridurre al minimo l'interazione fra il client e il server e lasciare la parte più intensiva del processo sul client; questo aspetto può essere riassunto nel seguente slogan: "XML fa lavorare Java".
Bisogna notare che la validazione dei dati in questi processi non è sempre necessaria; infatti la validazione dei dati è cruciale quando questi devono essere memorizzati in un database, ma non sempre questo è richiesto; per rendere questi processi il più efficienti possibile, XML permette che la validazione sia un optional in applicazioni dove non è necessaria.
L'esempio dei semiconduttori mostra come si integrano bene l'XML e Java in applicazioni in cui i dati devono essere manipolati in modo interessante sul client.

1.4.3 VISTE DIFFERENTI

Un utente può decidere di cambiare la visualizzazione dei dati senza dover scaricare differenti formati dal Web server. Una possibile applicazione di questa categoria è un dinamico tables of contents (TOC); è possibile attraverso un server Web, presentare all'utente il contenuto di una struttura dati utilizzando un TOC dinamico; con un click del

mouse su una parte del TOC, l'utente può ottenere livelli di dettaglio più specifici della struttura dati. Un TOC dinamico di questo tipo può essere generato a run-time direttamente dalla struttura gerarchica dei dati memorizzati in un database; sfortunatamente il ritardo intrinseco della rete Internet, rende il processo di espansione o di contrazione del TOC fastidioso per molti utenti. Una soluzione migliore è quella di scaricare l'intera struttura TOC sul Web client; quindi l'utente può espandere, contrarre, navigare nel TOC, supportato da processi molto veloci che girano direttamente sul suo client.
Un gruppo di studio alla Sun ha implementato questo tipo di soluzione nel Java-based HTML Help browser, ma le limitazioni dell'HTML richiedono al team ingegnose "capriole". In questa applicazione un TOC è costruito manualmente (le carenze dell'HTML rendono impossibile la generazione automatica del TOC direttamente dal documento), utilizzando un set di tag inventato per questo proposito. Quindi il TOC è inserito in un commento dentro la pagina HTML, per fare in modo che il Web browser non abbia problemi di riconoscimento del set di tag non convenzionale; un applet Java scaricato con il documento HTML interpreta il markup del TOC, fornendolo all'utente.
In pratica questa soluzione lavora molto bene; ma in ambiente XML, la creazione manuale del TOC non è necessaria; naturalmente attraverso un editor è necessario creare la struttura generale del TOC, ma il TOC specifico di una particolare struttura dati può essere generato a run-time e scaricato nel browser che lo visualizza grazie al Java applet.
La capacità di catturare e trasmettere le informazioni semantiche e strutturale dei dati, rende possibile attraverso l'XML, l'implementazione di una grande varietà di questo tipo di applicazioni; ad esempio:

- Con un semplice click del mouse si può optare per visualizzare la versione per macchine Sparc del manuale tecnico del sistema operativo Solaris, o la versione per macchine x86.
- Oppure si può optare per visualizzare il manuale in differenti lingue internazionali.
- Un documento che contiene molte annotazioni può essere visualizzato senza queste, oppure solo con le annotazioni, oppure sia con il testo che con le annotazioni, semplicemente attraverso un menu di selezione.
- Una agenda telefonica ordinata sul Cognome, può istantaneamente essere ordinata sul nome.

Questi sono solo alcuni esempi che possono essere implementati in ambiente Web grazie all'XML.

1.4.4 MOTORI DI RICERCA

XML permette di aggiungere informazioni semantiche al testo:

```
<Autore>Giancarlo Parma</Autore>
```

questo permette di semplificare la creazione di applicazioni che svolgono operazioni intelligenti con i documenti elettronici; un motore di ricerca sarebbe in grado di eseguire ricerche esplicite nel Web per trovare tutti i documenti in cui Giancarlo Parma è l'autore; in questo modo si può superare uno dei limiti dell'HTML, in cui i dati sono orientati al video e difficili da utilizzare per una elaborazione successiva; a questo riguardo, il commercio on-line è in pieno sviluppo e sempre più commercianti in tutto il mondo si stanno affacciando nel Web; però un'indagine su un campione di acquirenti abituali via Internet, ha

evidenziato una certa frustrazione da parte dei consumatori per la difficoltà di trovare i prodotti di cui hanno bisogno; il problema risiede nel sistema di indicizzazione delle merci, non sempre intuitivo e semplice come l'utente vorrebbe.
La chiave per risolvere questo tipo di problemi sta in questo slogan: "L'informazione ha bisogno di conoscere se stessa, ma ha anche bisogno di conoscere me"; supponiamo di dover implementare una guida TV personalizzata per un sistema via cavo di 500 canali; la guida TV personalizzata deve conoscere sia le preferenze e le caratteristiche dell'utente (livello di educazione, interessi, professione, età, etc.), sia le caratteristiche dei programmi trasmessi; queste informazioni devono essere fornite in modo tale da permettere al motore di ricerca implementato nella guida, di fare una selezione intelligente dei programmi più interessanti per l'utente; si ha quindi bisogno di un sistema standard che utilizzi uno specifico set di tag con cui poter esprimere le caratteristiche di un particolare programma (argomento, tipo di utenza a cui è rivolto, attori, lunghezza, data in cui è stato girato, lingua, etc.).
Questo è un semplice esempio che può naturalmente essere esteso ad un qualsiasi ambiente in cui l'informazione debba essere ritagliati sui gusti degli utenti; l'XML è un'ottima soluzione anche per questo tipo di problemi e permetterà ad applicazioni Web di competere realmente con la grande distribuzione dislocata sul territorio.

CAPITOLO 2. STRUTTURA E SINTASSI

Una delle caratteristiche principali dell'XML è la possibilità di fornire una struttura a un documento. Ogni documento XML comprende sia una *struttura logica* che una *struttura fisica*. La struttura logica è simile a un modello che indica quali elementi includere in un documento e in quale ordine. La struttura fisica contiene i dati effettivi utilizzati in un documento, quali il testo memorizzato nella memoria del computer, un'immagine memorizzata nel WWW e così via. Per comprendere la struttura di un documento XML, osserviamo questo modello:

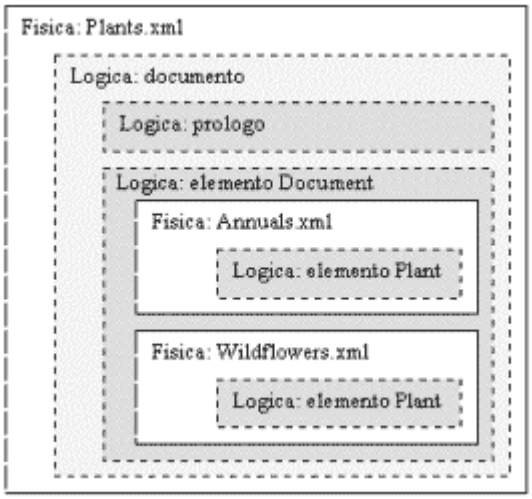


Fig 1.5:Un documento XML ha sia una struttura logica che una struttura fisica.

2.1 STRUTTURA LOGICA DEL LINGUAGGIO XML

La struttura logica fa riferimento all'organizzazione delle parti di un documento: in altre parole, indica il modo in cui viene creato un documento in contrapposizione al contenuto del documento stesso.
Un documento XML è costituito da dichiarazioni, elementi, istruzioni di elaborazione e commenti. Alcuni componenti sono opzionali, altri sono necessari.

PROLOGO
Il primo elemento strutturale di un documento XML è un *prologo* opzionale, costituito da due componenti principali anch'essi opzionali: la *dichiarazione XML* e la *dichiarazione del tipo di documento*.

DICHIARAZIONE XML La dichiarazione XML identifica la versione delle specifiche XML a cui è conforme il documento. Sebbene la dichiarazione XML sia un elemento opzionale, deve sempre essere inserita in documento XML. Il documento inizia con una dichiarazione XML di base:

```
<?xml version="1.0"?>
```

Una dichiarazione XML può inoltre contenere una *dichiarazione di codifica* (encoding) e una *dichiarazione di documento autonomo* (standalone). La dichiarazione di codifica identifica lo schema di codifica dei caratteri, ad esempio UTF-8 o EUC-JP. Schemi di codifica diversi assegnano formati di caratteri o linguaggi diversi. La dichiarazione di documento autonomo identifica l'esistenza delle dichiarazioni di markup esterne al documento. Questo tipo di dichiarazione può assumere valore *yes* o *no.*

DICHIARAZIONE DEL TIPO DI DOCUMENTO La dichiarazione del tipo di documento è costituita da codice di markup che indica le regole grammaticali o la definizione del tipo di documento DTD per una particolare classe di documenti. Questa dichiarazione può anche essere diretta a un file esterno che contiene tutta o parte della DTD e deve essere visualizzata dopo la dichiarazione XML e prima dell'elemento Document. Queste stringhe di codice aggiungono una dichiarazione del tipo di documento all'esempio:

```
<?xml version="1.0"?>
<!DOCTYPE Wildflowers SYSTEM "Wldflr.dtd">
```

L'ELEMENTO DOCUMENT
L'elemento Document contiene tutti i dati di un documento XML inclusi tutti i sottoelementi nidificati e le entità esterne. Può essere considerato simile all'unità C: del computer. Tutti i dati del computer sono memorizzati in questa singola unità in cui le cartelle e le sottocartelle contengono le singole parti di dati in una struttura logica e di semplice gestione. Queste stringhe di codice aggiungono un elemento Document, in questo caso l'elemento Plant all'esempio:

```
<?xml version="1.0"?>
<!DOCTYPE Wildflowers SYSTEM "Wldflr.dtd">

<PLANT>
 <COMMON>Columbine</COMMON>
 <BOTANICAL>Aquilegia canadensis</BOTANICAL>
</PLANT>
```

La ***nidificazione*** è il processo che consente di incorporare un oggetto o un costrutto l'uno all'interno dell'altro. Un documento XML può ad esempio contenere elementi nidificati e altri documenti. Ogni elemento secondario, cioè un elemento diverso dall'elemento Document risiede interamente all'interno del relativo elemento principale, così :

```
<DOCUMENT>
     <PARENT1>
          <CHILD1></CHILD1>
          <CHILD2></CHILD2>
     </PARENT1>
</DOCUMENT>
```

2.2 STRUTTURA FISICA DEL LINGUAGGIO XML

La struttura fisica di un documento XML è costituita da tutto il contenuto del documento stesso. Le unità di memorizzazione definite *entità*, possono essere parte integrante del

documento o possono essere esterne. Ogni entità è identificata da un nome univoco e da un contenuto specifico che può essere costituito da un singolo carattere all'interno del documento o da un file esterno di grandi dimensioni. In termini di struttura logica di un documento XML, le entità vengono dichiarate nel prologo e viene loro fatto riferimento nell'elemento Document.
Dopo aver dichiarato la DTD, l'entità può essere utilizzata in un punto qualsiasi del documento. Un riferimento di entità indica all'elaboratore di recuperare il contenuto di un'entità, come stabilito dalla dichiarazione di entità, e di utilizzarla all'interno del documento.

ENTITA' ANALIZZABILI E NON ANALIZZABILI
Un'entità può essere *analizzabile* o *non analizzabile*. Per entità analizzabile si intende un'entità in grado di essere letta dall'elaboratore di XML che ne consente l'estrazione. Al termine dell'estrazione, questo tipo di entità viene visualizzata come parte del testo del documento nella posizione di riferimento dell'entità stessa. Ad esempio, una dichiarazione del tipo analizzabile potrebbe essere questa :

```
<!ENTITY LR1 "light requirement: mostly shade">
```

Ogni volta che nel documento viene fatto riferimento a questa entità, quest'ultima verrà sostituita dal contenuto. Se si desidera modificare il contenuto dell'entità, è necessario effettuare questa operazione solo nella dichiarazione e la modifica si rifletterà in qualsiasi punto del documento in cui venga utilizzata l'entità.

RIFERIMENTI DI ENTITA' Il contenuto di ogni entità viene aggiunto al documento ogni volta che viene fatto riferimento a quell'entità. Il riferimento ha la funzione di segnaposto per l'autore del contenuto e l'elaboratore di XML colloca il contenuto effettivo nei punti di riferimento. Per includere un riferimento, bisogna inserire una e commerciale (*&*) e immettere il nome dell'entità seguito da punto e virgole (*;*). All'interno di un documento assumerebbe il seguente aspetto:

```
<TERM>Wild Ginger has the following &LR1;</TERM>
```

RIFERIMENTI DI ENTITA' DI PARAMETRO Un altro tipo di riferimento è quello relativo all'entità di parametro che utilizza un modulo (*%*) invece di una e commerciale anche se l'aspetto è simile a qualsiasi altro riferimento di entità. *%CDF*; è un esempio di entità di parametro.
Un'entità non analizzabile viene indicata talvolta come entità binaria in quanto il contenuto è spesso costituito da un file binario, ad esempio un'immagine, che non può essere interpretato direttamente dall'elaboratore XML. Un'entità non analizzabile richiede informazioni diverse da quelle incluse in un'entità analizzabile. Viene richiesta un'*annotazione* che identifica il formato o il tipo di risorsa per cui l'entità viene dichiarata. Ad esempio :

```
<!ENTITY MyImage SYSTEM "Image001.gif" NDATA GIF>
```

Questa dichiarazione significa che l'entità MyImage è un file binario nell'annotazione GIF. Perché queste dichiarazione di entità siano valide, anche l'annotazione deve essere dichiarata. La *dichiarazione di annotazione* consente all'applicazione di XML di gestire i file binari esterni. Nel caso dell'annotazione GIF utilizzata nell'esempio, può essere impiegata la dichiarazione di annotazione seguente:

```
<!NOTATION GIF SYSTEM "/Utils/Gifview.exe">
```

Questa stringa di codice indica all'elaboratore di XML di utilizzare Gifview.exe per elaborare l'entità di tipo *GIF* ogni volta che viene rilevata. Dopo essere stata dichiarata, la dichiarazione di annotazione può essere utilizzata all'interno del documento.

ENTITA' PREDEFINITE

Nel linguaggio XML alcuni caratteri sono utilizzati per contrassegnare il documento in modo specifico. Le parentesi angolari (<>) e la barra (/) sono interpretate come markup e non come dati di un carattere effettivo:

<PLANT>Blodroot</PLANT>

Questi e altri caratteri sono riservati per il markup e non possono essere utilizzati come contenuto. Se si desidera che questi caratteri siano visualizzati come dati, è necessario utilizzare determinati codici:

<	< (parentesi angolare di apertura)
>	> (parentesi angolare di chiusura)
&	& (e commerciale)
'	‘ (apostrofo)
"	" (virgolette doppie)

ENTITA' INTERNE ED ESTERNE

Nel primo caso si tratta di un'entità in cui non esistono unità di memorizzazione fisica separate e il cui contenuto viene fornito nella dichiarazione corrispondente, ad esempio:

```
<!ENTITY                                                    LR1
 "light requirement: mostly shade">
```

Un'entità esterna fa riferimento a un'unità di memorizzazione nella dichiarazione mediante un identificatore pubblico o di sistema. L'identificatore di sistema fornisce un collegamento alla posizione in cui si trova il contenuto dell'entità, ad esempio un URI (Uniform Resource Identifier) come ad esempio:

```
<!ENTITY MyImage
          SYSTEM
 "http://www.wildflowers.com/Image001.gif" NDATA GIF>
```

In questo caso l'elaboratore di XML deve necessariamente leggere il file Image001.gif per recuperare il contenuto di questa entità.

Oltre all'identificatore di sistema, l'entità può includere un identificatore pubblico che fornisce un metodo opzionale e alternativo per il recupero del contenuto di un'entità da parte dell'elaboratore di XML. Questo identificatore può essere ad esempio utilizzato se l'applicazione è collegata a una libreria del documento disponibile pubblicamente. Se l'elaboratore non è in grado di generare una posizione appropriata per l'identificatore pubblico, è necessario che venga controllato l'URI specificato dall'identificatore di sistema. Ad esempio:

```
<!ENTITY                        MyImage                      PUBLIC
 "-//Wildflowers/TEXT Standard images//EN"
 "http://www.wildflowers.com/Image001.gif"
```

```
NDATA GIF>
```

L'elaboratore di XML verifica l'identificatore pubblico all'interno di un elenco di risorse a cui è collegato e scoprire che non è necessaria una nuova copia dell'entità perché già disponibile localmente.

2.3 SINTASSI XML

Le regole strutturali del linguaggio XML si riflettono nelle regole linguistiche o *sintassi*.

2.3.1 APERTURA E CHIUSURA DEI TAG

Nel codice HTML un elemento contiene in genere sia tag di apertura che di chiusura. A differenza dell'HTML, l'XML richiede che un tag di chiusura venga utilizzato per ogni elemento.
Si consideri ad esempio l'elemento HTML Paragraph che dovrebbe in genere includere un tag di apertura, il contenuto e un tag di chiusura come mostrato di seguito:

```
<P>Questo è un elemento HTML Paragraph.</P>
```

Non sempre viene utilizzato un tag di chiusura in questo contesto. Questo avviene perché l'HTML e il linguaggio di origine SGML consentono di omettere i tag di chiusura senza invalidare il codice.
Poiché un paragrafo in HTML non può essere annidato all'interno di un altro paragrafo, l'elaboratore è in grado di leggere il tag di apertura del paragrafo e di presumere che indichi anche la fine del paragrafo precedente. Queste tecniche di minimizzazione non sono consentite nel linguaggio XML. Questa caratteristica costituisce la differenza sintattica più evidente tra i due linguaggi.

IL TAG DI ELEMENTO VUOTO
Il linguaggio XML supporta un collegamento per elementi vuoti, il *tag di elemento vuoto*. Questo tag unisce i tag di apertura e di chiusura per un elemento senza alcun contenuto. Viene utilizzato un formato speciale: <NOMETAG/>. In questo caso la barra segue il nome del tag, il che non è possibile nel linguaggio HTML.

2.3.2 ATTRIBUTI

Gli *attributi* consentono di associare valori a un elemento senza che siano considerati parte del contenuto dell'elemento stesso. Ad esempio osserviamo un comune elemento HTML e l'utilizzo di un attributo:

```
<A HREF = "http://www.microsoft.com">Microsoft Home Page</A>
```

In questo caso l'elemento Anchor indicato dal tag <A> contiene un attributo denominato HREF. Il valore dell'attributo è http://www.microsoft.com. Mentre il valore non viene mai visualizzato dall'utente, l'attributo contiene importanti informazioni relative all'elemento e

fornisce la destinazione dell'ancoraggio. Questo formato del nome e del valore mostra il modo in cui sono utilizzati gli attributi nel linguaggio XML.
Questo esempio aggiunge un attributo a uno degli elementi del documento:

```
<?xml version="1.0"?>
 <!DOCTYPE Wildflowers SYSTEM "Wldflr.dtd">

 <PLANT ZONE=3>
  <COMMON>Columbine</COMMON>
  <BOTANICAL>Aquilegia canadensis</BOTANICAL>
 </PLANT>
```

L'attributo ZONE del tag di apertura <PLANT> segue il formato del nome e del valore.

2.4 DOCUMENTI XML VALIDI E BEN FORMATI

Il linguaggio XML possiede due caratteristiche fondamentali, la capacità di fornire una struttura ai documenti e di rendere i dati autodescrittivi. Queste caratteristiche non sarebbero di alcuna utilità se non si potessero far rispettare le regole strutturali e grammaticali.

2.4.1 DOCUMENTI VALIDI

La definizione del tipo di documento DTD specificata nel prologo delinea tutte le regole relative a un documento. Un documento XML *valido* segue tutte queste regole rigidamente. Un documento valido è conforme anche a tutti i limiti di validità identificati dalle specifiche relative all'XML.

L'elaboratore dovrà comprendere i limiti di validità delle specifiche XML e verificare possibili violazioni all'interno del documento. Se l'elaboratore trova un errore, deve comunicarlo all'applicazione XML. Dovrà inoltre leggere la DTD, convalidare il documento e riportare qualsiasi violazione all'applicazione XML. Dato che questi controlli possono richiedere tempo e occupare larghezza di banda e poiché la convalida non sempre è necessaria, il linguaggio XML supporta la nozione di documento ben formato.

2.4.2 DOCUMENTI BEN FORMATI

Anche se *ben formato* significa che è necessario seguire alcune regole, non è richiesto la stessa rigidità dei limiti di validità. Il concetto di documento ben formato è relativamente nuovo in XML. Un documento XML ben formato è più facile da leggere per un programma ed è pronto per la distribuzione in rete. Più specificatamente, i documenti ben formati hanno queste caratteristiche:

- Tutti i tag di apertura e di chiusura corrispondono.
- I tag vuoti utilizzano una sintassi XML speciale.

- Tutti i valori degli attributi sono racchiusi tra virgolette.

- Tutte le entità sono dichiarate.

Quindi, un documento XML valido rispetta i tag e le norme di nidificazione impostate nel DTD del documento, mentre un documento XML ben formato viene strutturato in modo appropriato per l'utilizzo da parte di un computer.

CAPITOLO 3. DEFINIZIONE DEL TIPO DI DOCUMENTO (DTD)

Un documento XML comprende, come sappiamo, il prologo che contiene la dichiarazione XML e la dichiarazione del tipo di documento, che identifica il tipo di documento specifico elaborato e le regole che controllano il documento completo. Queste regole sono denominate definizione del tipo di documento o DTD e costituiscono la parte più complessa della dichiarazione del tipo di documento.

3.1 STRUTTURA DELLA DTD

Una DTD può essere costituita da due parti: un *sottoinsieme DTD esterno* e un *sottoinsieme DTD interno*. Nel primo caso si tratta di una DTD che esiste all'esterno del contenuto del documento, nel secondo caso si tratta di una DTD inclusa all'interno del documento XML. Un documento può contenere una o entrambi i tipi di sottoinsiemi. In questo caso il sottoinsieme interno viene elaborato per primo e gli viene data la precedenza su qualsiasi sottoinsieme esterno. Questa funzionalità è utile agli autori che impiegano una DTD esterna, ma che desiderano personalizzare alcune parti della DTD per un'applicazione specifica.
Se si desidera includere un sottoinsieme DTD interno al documento, è sufficiente scriverlo nella dichiarazione del tipo di documento. Un sottoinsieme DTD esterno tuttavia deve essere incluso mediante un riferimento DTD, che indica al processore dove trovare il sottoinsieme esterno specificando il nome del file DTD. Il riferimento DTD contiene inoltre informazioni relative all'autore, all'obiettivo e al linguaggio utilizzato nella DTD. Ad esempio:

```
<!DOCTYPE catalog PUBLIC "-//flowers//DTD   Standard //EN"
 http://www.wildflowers.com/dtd/Wldflr.dtd>
```

3.2 CREAZIONE DI UNA DTD SEMPLICE

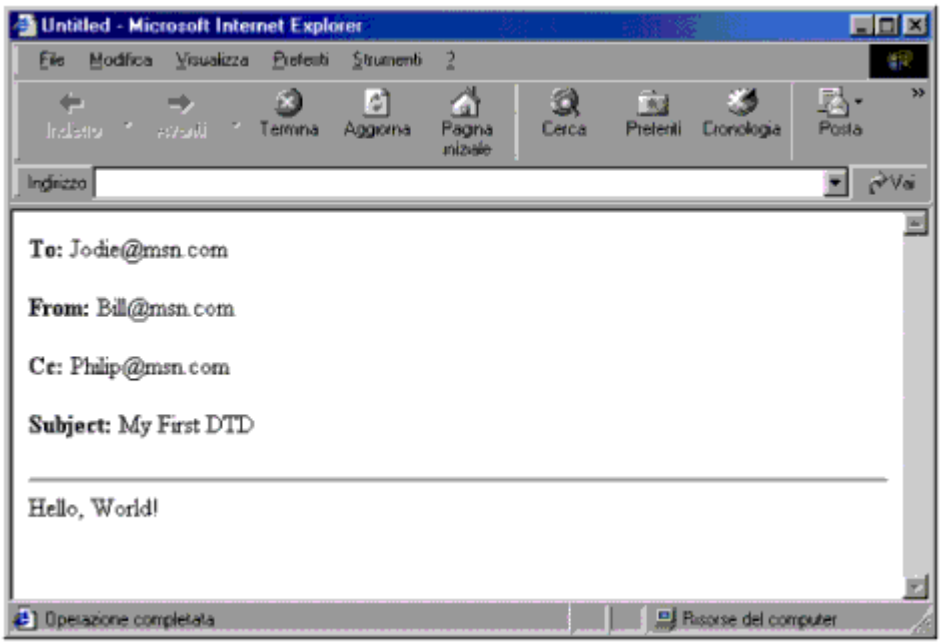
Creiamo un documento XML di messaggio di posta elettronica con un sottoinsieme DTD interno.

```
<?xml version="1.0"?>

 <!DOCTYPE EMAIL [
  <!ELEMENT EMAIL (TO, FROM, CC, SUBJECT, BODY)>
  <!ELEMENT TO (#PCDATA)>
  <!ELEMENT FROM (#PCDATA)>
  <!ELEMENT CC (#PCDATA)>
  <!ELEMENT SUBJECT (#PCDATA)>
  <!ELEMENT BODY (#PCDATA)>
 ]>
```

```
<EMAIL>
 <TO>Jodie@msn.com</TO>
 <FROM>Bill@msn.com</FROM>
 <CC>Philip@msn.com</CC>
 <SUBJECT>My first DTD</SUBJECT>
 <BODY>Hello, World</BODY>
</EMAIL>
```

Si noti che il codice contiene informazioni aggiuntive nella dichiarazione del tipo di documento. In questo caso si tratta del sottoinsieme DTD interno e identifica gli elementi che possono essere presenti nel documento e il tipo di dati che deve contenere. Se questo documento viene eseguito visualizzando la pagina XML, il documento avrà questo aspetto:



In questo caso l'elaboratore sta convalidando il documento in base alla DTD. In altri termini la pagina XML che visualizza il documento esegue l'analisi utilizzando un elaboratore di convalida. Ciò significa che l'elaboratore verifica il documento in base alle regole della DTD per accertare che tutto il codice utilizzato nel documento sia valido.
Per verificare il funzionamento della convalida, aggiungiamo un elemento non incluso nella DTD al documento, ad esempio un elemento Signature, al termine del documento, come:

```
<?xml version="1.0"?>

 <!DOCTYPE EMAIL [
  <!ELEMENT EMAIL (TO, FROM, CC, SUBJECT, BODY)>
   <!ELEMENT TO (#PCDATA)>
   <!ELEMENT FROM (#PCDATA)>
   <!ELEMENT CC (#PCDATA)>
   <!ELEMENT SUBJECT (#PCDATA)>
   <!ELEMENT BODY (#PCDATA)>
 ]>
 <EMAIL>
  <TO>Jodie@msn.com</TO>
```

```
  <FROM>Bill@msn.com</FROM>
  <CC>Philip@msn.com</CC>
  <SUBJECT>My first DTD</SUBJECT>
  <BODY>Hello, World</BODY>
  <SIGNATURE>Bill</SIGNATURE>
 </EMAIL>
```

Se si cerca di eseguire questo documento, verrà visualizzato un messaggio di errore perché l'elaboratore non trova la dichiarazione relativa all'elemento Signature nella DTD simile a questo:

Element content is invalid according to the DTD/Schema.
(Il contenuto dell'elemento non è valido in base alla DTD/Schema.)

Modifichiamo adesso una parte della DTD. Notiamo che la prima dichiarazione dell'elemento è relativa all'elemento Email:

```
<!ELEMENT EMAIL (TO,FROM,CC,SUBJECT,BODY)>
```

All'interno di questa riga di codice tra le parentesi è presente un elenco degli altri elementi che possono essere contenuti nel documento. Questo elenco viene definito *modello di contenuto* e identifica gli elementi secondari che l'elemento Email deve contenere e l'ordine in cui sono elencati.
Ad esempio rimuoviamo l'elemento Subject dal modello di contenuto.

```
<!DOCTYPE EMAIL [
 <!ELEMENT EMAIL (TO, FROM, CC, BODY)>
 <!ELEMENT TO (#PCDATA)>
 <!ELEMENT FROM (#PCDATA)>
 <!ELEMENT CC (#PCDATA)>
 <!ELEMENT SUBJECT (#PCDATA)>
 <!ELEMENT BODY (#PCDATA)>
]>
```

L'esecuzione del documento causerebbe un errore, perché il documento non ha seguito il modello di documento specificato. Torniamo al documento originale per modificare l'ordine degli elementi From e Cc affinché la parte superiore del documento assuma l'aspetto seguente:

```
<EMAIL>
 <TO>Jodie@msn.com</TO>
 <CC>Philip@msn.com</CC>
 <FROM>Bill@msn.com</FROM>
 <SUBJECT>My first DTD</SUBJECT>
 <BODY>Hello, World</BODY>
 <SIGNATURE>Bill</SIGNATURE>
</EMAIL>
```

Ancora una volta al momento dell'esecuzione del documento, verrà restituito un errore perché l'elaboratore prevedeva un altro elemento al posto di quello ottenuto.
E' dunque chiaro che la DTD ha il ruolo di un rigido regolamento per i documenti XML, quindi è importante fare attenzione al momento della creazione delle DTD.

3.3 DICHIARAZIONI DI ELEMENTI

Ogni dichiarazione di elemento contiene il nome dell'elemento e il tipo di dati definito *specifiche di contenuto* costituite da uno tra i quattro tipi seguenti:

- Un elenco di altri elementi, denominato modello di contenuto
- La parola chiave EMPTY
- La parola chiave ANY
- Contenuto di vario tipo

ULTERIORI INFORMAZIONI SUL MODELLO DI CONTENUTO
La DTD dell'esempio precedente iniziava con una dichiarazione di elemento inclusa nel modello di contenuto, come illustrato nella parentesi che segue:

```
<!ELEMENT EMAIL (TO, FROM, CC, SUBJECT, BODY)>
```

L'elemento Email contiene solo sottoelementi o elementi secondari. Per ogni elemento del modello di contenuto deve essere visualizzata una dichiarazione di elemento corrispondente nella parte restante della DTD che segue.

DICHIARAZIONE DI ELEMENTO VUOTO
Per dichiarare che un elemento non può avere alcun contenuto, è possibile utilizzare la parola chiave EMPTY nella dichiarazione di elemento, come indicato di seguito:

```
<!ELEMENT TEST EMPTY>
```

Un elemento Test di un documento che includa la dichiarazione precedente non potrebbe mai avere alcun contenuto e sarebbe necessario che fosse indicato come elemento vuoto, ad esempio <TEST/>. Anche se gli elementi vuoti potrebbero sembrare inutili, possono contenere attributi in grado di fornire un contenuto significativo o di funzioni specifiche all'interno di un documento. Il tag
 in HTML è un esempio di tag di elemento vuoto.

DICHIARAZIONE DI TUTTI GLI ELEMENTI
La specifica di contenuto ANY è esattamente l'opposto della parola chiave precedente. Se una dichiarazione di elemento utilizza la parola chiave ANY per le specifiche di contenuto, quel tipo di elemento potrà avere qualsiasi tipo di contenuto in base alle disposizioni della DTD, disposto in un ordine qualsiasi. La dichiarazione di tutti gli elementi assume questo aspetto:

```
<!ELEMENT TEST ANY>
```

CONTENUTO DI VARIO TIPO
Le specifiche di contenuto possono anche essere costituite da un singolo insieme di alternative separate dal simbolo pipe (|). Ad esempio:

```
<!ELEMENT EXAMPLE (#PCDATA|x|y|z)*>
```

3.4 TIPI DI DATI

All'interno del contenuto dei documenti, il linguaggio XML consente di utilizzare dati di caratteri analizzabili dichiarati mediante la parola chiave #PCDATA e i dati di caratteri dichiarati mediante la parola chiave CDATA. I dati di caratteri analizzabili sono dati di caratteri di markup, contengono quindi tag di markup. I dati di caratteri sono costituiti da testo ordinario che può includere caratteri in genere riservati al markup. In base all'impostazione predefinita, gli elaboratori di XML presuppongono che il contenuto di un file XML sia costituito da dati di caratteri.
Mentre i dati di caratteri analizzabili sono in genere utilizzati nel contenuto di un documento XML, i dati di carattere possono essere utilizzati nel caso in cui un autore desideri includere dati che non possono essere analizzati. Per dichiarare una sezione come dati di carattere, è necessario indicare l'inizio della sezione con la sequenza <![CDATA[e la fine con due parentesi di chiusura]]. Tutti i dati che risiedono all'interno di questo insieme di marcatori verranno interpretati come semplici dati non analizzabili.

3.5 SIMBOLI RELATIVI ALLA STRUTTURA

Il linguaggio XML utilizza una serie di simboli per specificare la struttura di una dichiarazione di elementi. La tabella seguente identifica i simboli disponibili, lo scopo di ogni simbolo, un esempio di come vengono utilizzati e il loro significato.

Simbolo	**Scopo**	**Esempio**	**Significato**
Parentesi	Racchiudono una sequenza, un gruppo di elementi o una serie di alternative	*(content1, content2)*	L'elemento deve contenere la sequenza *content1* e *content2.*
Virgola	Separa gli elementi di una sequenza e identifica l'ordine in cui devono essere visualizzati	*(content1, content2, content3)*	L'elemento deve contenere *content1*, *content2* e *content3* nell'ordine specificato.
Pipe	Separa gli elementi in un gruppo di alternative	*(content1*| *content2*| *content3)*	L'elemento deve contenere *content1*, *content2* o *content3*.
Punto di domanda	Indica che un elemento deve essere visualizzato una sola volta o non apparire mai	*content1?*	L'elemento può contenere *content1*. Se *content1* viene visualizzato, deve apparire una sola volta.

Asterisco	Indica che l'elemento può essere visualizzato ogni volta che l'autore desidera	*content1**	L'elemento può contenere *content1*. Se viene visualizzato, può apparire una o più volte.
Segno più	Indica che un elemento deve essere visualizzato una o più volte	*content1+*	L'elemento deve contenere *content1* una volta, ma può essere visualizzato anche più di una volta.
Nessun simbolo	Indica che deve essere visualizzato un elemento	*content1*	L'elemento deve contenere *content1*.

3.6 ATTRIBUTI

Oltre alla definizione della struttura di un elemento e al tipo di contenuto, è possibile associare attributi a un elemento. Gli attributi forniscono informazioni aggiuntive relative all'elemento o al contenuto dell'elemento.

DICHIARAZIONI DI ATTRIBUTO
Nel linguaggio XML gli attributi vengono dichiarati nella DTD utilizzando la sintassi seguente:

```
<!ATTLIST ElementName AttributeName Type  Default>
```

In questo caso <!ATTLIST> rappresenta il tag che identifica una dichiarazione di attributo. La voce *ElementName* rappresenta il nome dell'elemento a cui vengono applicati gli attributi, La voce *AttributeName* rappresenta il nome dell'attributo. La voce *Type* identifica il tipo di attributo dichiarato. La voce *Default* specifica le impostazioni predefinite relative all'attributo.
Ecco elencati i tipi di attributi disponibili per il linguaggio XML:

Tipo di attributo	**Utilizzo**
CDATA	In questo attributo possono essere utilizzati solo dati in formato carattere.
ENTITY	Il valore dell'attributo deve fare riferimento a un'entità binaria esterna dichiarata nella DTD.
ENTITIES	E' equivalente all'attributo ENTITY, ma consente l'utilizzo di più valori separati da spazi.

ID	Il valore dell'attributo deve essere un identificatore univoco. Se un documento contiene attributi ID con lo stesso valore, l'elaboratore produrrà un errore.
IDREF	Il valore deve essere un riferimento a un ID dichiarato in un altro punto del documento. Se l'attributo non corrisponde al valore dell'ID specificato, l'elaboratore produrrà un errore.
IDREFS	E' equivalente all'attributo IDREF, ma consente l'utilizzo di più valori separati da spazi.
NMTOKEN	Il valore dell'attributo consiste in una qualsiasi combinazione di caratteri del token del nome, rappresentati da lettere, numeri, punti trattini, due punti o caratteri di sottolineatura.
NMTOKENS	E' equivalente all'attributo NMTOKEN, ma consente l'utilizzo di più valori separati da spazi.
NOTATION	Il valore dell'attributo deve fare un riferimento a un'annotazione dichiarata in un altro punto della DTD. La dichiarazione può anche essere costituita da un elenco di annotazioni. Il valore deve corrispondere a una delle annotazioni dell'elenco. Ogni annotazione deve avere la relativa dichiarazione nella DTD.
Enumerated	Il valore dell'attributo deve corrispondere a uno dei valori inclusi. Ad esempio: <!ATTLIST *MyAttribute* (*content1*|*content2*)>.

La parte finale della dichiarazione di attributo è l'impostazione predefinita per il valore dell'attributo. Le impostazioni predefinite per i quattro tipi sono:

Impostazione predefinita	**Utilizzo**
#REQUIRED	Ogni elemento contenente questo attributo deve specificarne un valore. Un valore mancante può causare un errore.
#IMPLIED	Questo attributo è opzionale. L'elaboratore può ignorare questo attributo se non viene rilevato alcun valore.
#FIXED *fixedvalue*	Questo attributo deve avere il valore *fixedvalue*. Se l'attributo non è incluso nell'elemento, viene stabilito il valore *fixedvalue*.

Default	Identifica un valore predefinito per un attributo. Se l'elemento non include l'attributo, viene stabilito il valore *default*.

Nel documento d'esempio mostriamo l'utilizzo degli attributi aggiungendo alcune dichiarazioni di attributo alla DTD:

```
<?xml version="1.0"?>

<!DOCTYPE EMAIL [
 <!ELEMENT EMAIL (TO+, FROM, CC*, BCC*, SUBJECT?, BODY?)>
 <!ATTLIST EMAIL
  LANGUAGE(Western|Greek|Latin|Universal) " Western"

  ENCRYPTED CDATA #IMPLIED
  PRIORITY (NORMAL|LOW|HIGH) "NORMAL">

 <!ELEMENT TO (#PCDATA)>
 <!ELEMENT FROM (#PCDATA)>
 <!ELEMENT CC (#PCDATA)>

 <!ELEMENT BCC (#PCDATA)>
 <!ATTLIST BCC
  HIDDEN CDATA #FIXED "TRUE">

 <!ELEMENT SUBJECT (#PCDATA)>
 <!ELEMENT BODY (#PCDATA)>
]>
```

In questo esempio sono stati aggiunti attributi all'elemento Email e al nuovo elemento Bcc. Il primo attributo aggiunto all'elemento Email è *LANGUAGE*. Questo attributo può contenere una tra le numerose opzioni. L'attributo conterrà il valore predefinito *Western* se non verrà specificato un altro valore. L'attributo successivo dell'elemento Email è *ENCRYPTED*. Questo elemento deve contenere i dati di carattere e poiché l'impostazione predefinita è *#IMPLIED*, l'elaboratore ignorerà questo attributo se non verrà specificato alcun valore. L'ultimo attributo dell'elemento Email è *PRIORITY*. Questo attributo può assumere uno dei tre valori *NORMAL*, *LOW* e *HIGH*. Il valore predefinito è *NORMAL*.
L'attributo *HIDDEN* è stato incluso nell'elemento Bcc. Questo attributo è di tipo *CDATA* e il valore predefinito di *#FIXED* viene specificato dopo la parola chiave *#FIXED*. Questo attributo deve sempre specificare il valore nella DTD, in questo caso *TRUE*.

3.7 ENTITÀ

Oltre all'entità generale viste nel capitolo precedente esiste un altro tipo di entità definita *entità di parametro*.
La maggior parte delle entità deve essere dichiarata nella DTD. Alcune entità predefinite sono incorporate nel codice XML e sono utilizzate per visualizzare i caratteri generalmente impiegati per il markup. Le dichiarazioni di entità seguono la stessa sintassi di base utilizzata dalle altre dichiarazioni:

```
<!ENTITY EntityName EntityDefinition>
```

Le entità della DTD possono essere analizzabili o non analizzabili. Le entità del primo tipo o entità di testo contengono testo che farà parte del documento XML. Le entità del secondo tipo o entità binarie sono in genere riferimenti a un file binario esterno. Le entità non analizzabili possono anche essere costituite da testo non analizzabile ed è quindi preferibile pensare a queste entità come elementi che non sono stati creati per essere considerati parte del codice XML.

ENTITA' INTERNE
Le entità interne vengono dichiarate nella DTD e includono il contenuto che verrà utilizzato nel documento. Ad esempio, questa riga di codice aggiunge un'entità interna definita *SIGNATURE*:

```
<!ENTITY SIGNATURE "Bill">
```

Ogni volta che viene fatto riferimento all'entità nel documento, *&SIGNATURE*, quest'ultima verrà sostituita dal relativo contenuto, in questo caso *Bill*.

ENTITA' ESTERNE: PAROLE CHIAVE SYSTEM E PUBLIC
Le entità esterne fanno riferimento a file esterni, anche ad altri file XML. Ad esempio, questa entità fa riferimento a un file GIF esterno e verrà visualizzata nel corpo del documento XML:

```
<!ENTITY IMAGE1 SYSTEM "Xmlquot.gif" NDATA GIF>
```

Una dichiarazione di entità esterna può includere la parola chiave *SYSTEM* o *PUBLIC*. Molte DTD sono sviluppate localmente, vengono cioè sviluppate per un'azienda o organizzazione specifica o per un sito Web particolare. In questo caso andrebbe utilizzata la parola chiave *SYSTEM*. Questa parola chiave è seguita da un URI (Uniform Resource Identifier) che indica all'elaboratore dove reperire l'oggetto indicato nella dichiarazione. Nell'esempio precedente, il nome di file era utilizzato perché il codice aveva impiego locale. Nella dichiarazione che segue, l'URI è un indirizzo Web che collega alla posizione dei file di riferimento:

```
<!ENTITY IMAGE1 SYSTEM http://XMLCo.com/Images/Xmlquot.gif NDATA
GIF>
```

Alcune DTD sono standard stabiliti disponibili per un'ampia gamma di utenti. Sarebbe necessario utilizzare la parola chiave *PUBLIC*, seguita dall'identificatore pubblico che l'elaboratore può impiegare se è disponibile una libreria standard. Dopo l'identificatore pubblico è inserito un URI, simile a quello utilizzato con la parola chiave *SYSTEM*. Un esempio potrebbe essere questo:

```
<!ENTITY IMAGE1 PUBLIC "-//XMLCo//TEXT Standard Images//EN"
 "http://XMLCo.com/Images/Xmlquot.gif" NDATA   GIF>
```

ENTITA' ESTERNE: ANNOTAZIONI E DICHIARAZIONI DI ANNOTAZIONI
Consideriamo la dichiarazione di entità:

```
<!ENTITY IMAGE1 SYSTEM "Xmlquot.gif" NDATA GIF>
```

Un'*annotazione* NDATA GIF viene visualizzata nella parte finale della dichiarazione. Questa annotazione indica all'elaboratore il tipo di oggetto a cui viene fatto riferimento. A questo punto se viene semplicemente aggiunta la dichiarazione di entità alla DTD e viene eseguita attraverso l'elaboratore, verrà visualizzato un messaggio di errore simile al seguente:
Declaration 'IMAGE1' contains reference to undefined notation 'GIF'.
(La dichiarazione 'IMAGE1' contiene un riferimento a un'annotazione 'GIF' non identificata.)
L'errore si verifica perché la dichiarazione di entità fa riferimento a un tipo di file binario e all'elaboratore non è stato indicato come operare con questo file. Si tratta di un'entità non analizzabile che l'elaboratore non è in grado di comprendere. In questo caso l'annotazione deve essere dichiarata come *dichiarazione di annotazione*. Una dichiarazione di annotazione indica all'elaboratore come operare con un tipo di file binario specifico. Le dichiarazioni di annotazione hanno il seguente formato:

```
<!NOTATION GIF SYSTEM "Iexplore.exe">
```

Questa dichiarazione indica all'elaboratore di utilizzare il programma Iexplore.exe per elaborare il file GIF ogni volta che nella DTD ne viene rilevato uno.

ENTITA' DI PARAMETRO
Anche se le entità di parametro funzionano in modo simile alle entità generali, possiedono un'importante differenza sintattica. Le entità di parametro utilizzano il simbolo di percentuale (%) nelle dichiarazioni e nei riferimenti. Nella dichiarazione di entità il simbolo di percentuale segue la parola chiave *!ENTITY*, ma precede il nome dell'entità come illustrato di seguito. Si noti che è richiesto uno spazio singolo prima e dopo il simbolo &:

```
<!ENTITY % ENCRYPTION
 "40bit CDATA #IMPLIED
 128bit CDATA #IMPLIED">
```

E' ora possibile fare riferimento a questa entità in un altro punto della DTD. Ad esempio:

```
<!ELEMENT EMAIL (TO+, FROM, CC*, BCC*,  SUBJECT?, BODY?)>
<!ATTLIST EMAIL
 LANGUAGE(Western|Greek|Latin|Universal)  "Western"
 ENCRYPTED %ENCRYPTION;
 PRIORITY (NORMAL|LOW|HIGH) "NORMAL">
```

Il riferimento all'entità di parametro &ENCRYPTION; utilizza lo stesso formato di base del riferimento di entità generale, a eccezione del simbolo % che sostituisce il simbolo &.
Le entità di parametro possono rivelarsi un metodo utile per creare uno stile personale all'interno delle DTD e rendere queste ultime più concise e meglio organizzate. Tuttavia queste entità dovrebbero essere utilizzate con cautela, dato che possono creare situazioni complesse all'interno di un documento in grado di rendere difficoltosa la gestione.

3.8 LE PAROLE CHIAVE IGNORE E INCLUDE

Le parole chiave *IGNORE* e *INCLUDE* possono essere utilizzate dagli autori per "attivare" o "disattivare" porzioni della DTD. *IGNORE* e *INCLUDE* sono utilizzate nella DTD per creare all'interno del documento condizioni adatte a vari scopi. L'utilizzo di *IGNORE* e *INCLUDE* consente ad esempio di verificare diverse strutture durante il controllo delle variazioni. *IGNORE* e *INCLUDE* sono utilizzati in modo simile a *CDATA*:

```
<![IGNORE [DTD section]]>
<![INCLUDE [DTD section]]>
```

Nessuna parola chiave può apparire all'interno di una dichiarazione e ogni *sezione della DTD* deve includere una dichiarazione completa o una serie di dichiarazioni, commenti e spazi vuoti. Vediamo un esempio dell'utilizzo delle parole chiave:

```
<![IGNORE[<!ELEMENT BCC (#PCDATA)>
<!ATTLIST BCC
 HIDDEN CDATA #FIXED "TRUE">]]>
<![INCLUDE[<!ELEMENT SUBJECT (#PCDATA)>]]>
```

Questo frammento di codice indica all'elaboratore di ignorare l'elemento Bcc e l'elenco di attributi e includere l'elemento Subject.

3.9 ISTRUZIONI DI ELABORAZIONE.

Le *istruzioni di elaborazione* (PI, Processing Instructions) forniscono indicazioni all'applicazione che elabora il documento. Queste istruzioni vengono in genere visualizzate nel prologo, ma possono essere posizionate in un punto qualsiasi del documento XML. L'istruzione di elaborazione più comune è la dichiarazione XML inclusa nella parte superiore del documento di esempio:

```
<?xml version=1.0"?>
```

Le istruzioni di elaborazione sono scritte con la sequenza *<?*, seguita dal nome dell'istruzione, da un valore o da un'istruzione e sono chiuse con *?>*. Il nome o la *destinazione di PI* identifica quale applicazione dovrebbe seguire le istruzioni. Esempi di istruzioni di elaborazione sono:

```
<?AVI CODEC="VIDEO1" COLORS="256"?>
<?WAV COMPRESSOR="ADPCM" BITS="8"  RESOLUTION="16"?>
```

3.10 COMMENTI

I commenti rappresentano una delle parti generiche della DTD. Anche se i commenti non sono necessari, vengono ampiamente utilizzati per migliorare la leggibilità di un documento. E' possibile aggiungere commenti per spiegare lo scopo di una determinata sezione della DTD, per indicare il significato dei riferimenti e per altri obiettivi. I commenti si rivelano utili come promemoria durante le fasi della codifica, se si verifica la necessità di tornare alla DTD ed effettuare le modifiche oppure se un altro autore utilizza il documento.

I commenti non sono vincolati alla DTD e possono essere utilizzati in tutto il documento. Dato che i commenti sono a vantaggio esclusivo del lettore, qualsiasi elaboratore di XML ne ignorerà la presenza. I commenti vengono visualizzati tra tag di commento *(<!-- -->*) e possono includere qualsiasi combinazione di testo, markup e simboli a eccezione di combinazioni di simboli che costituiscono i tag di commento.

3.11 DTD ESTERNE

E' possibile separare documenti e DTD per semplificarne l'utilizzo. Dopo aver creato una DTD separata, è possibile farvi riferimento all'interno di qualsiasi documento.
Per separare una parte della DTD nel documento XML basta tagliare semplicemente la porzione di DTD e incollarla nel nuovo file di testo. Il nuovo nome di file dovrebbe avere estensione *.dtd*.
La separazione della DTD dal documento riduce notevolmente la dimensione del file del documento XML e fornisce altri vantaggi. Dato che ora la DTD è un file separato, può essere utilizzata in altri documenti da chiunque vi abbia accesso. Un altro autore può creare un documento utilizzando la stessa struttura con un contenuto completamente diverso. Poiché il nuovo documento seguirebbe la DTD, potrebbe essere letto da qualsiasi applicazione in grado di elaborare la DTD.

3.12 VOCABOLARI

Un *vocabolario XML* è un insieme di elementi e della struttura di un tipo di documento specifico. I vocabolari sono definiti in una DTD che rappresenta il regolamento per quel vocabolario. I vocabolari sono utilizzati correntemente su Internet e in alcune organizzazioni e aziende. Uno dei vocabolari principali e maggiormente noti è il Channel Definition Format (CDF) impiegato per definire le pagine Web progettate per essere inviate automaticamente agli utenti client.
I vocabolari sono adatti per le applicazioni verticali e per lo sviluppo di sistemi di interscambio di dati per aziende specifiche, ad esempio telecomunicazioni, prodotti farmaceutici e istituzioni legali.

CHANNEL DEFINITION FORMAT
Channel Definition Format (CDF) viene utilizzato per descrivere il comportamento delle pagine Web in un modello di invio automatico. CDF viene utilizzato da Microsoft Internet Explorer e descrive i processi quali pianificazioni di download, visualizzazione della barra dei canali, utilizzo delle pagine e frequenza degli aggiornamenti.

OPEN FINANCIAL EXCHANGE
Open Financial Exchange (OFX) è correntemente un'applicazione SGML utilizzata da pacchetti software per comunicare con le istituzioni finanziarie. OFX sarà presto basato sul linguaggio XML.

OPEN SOFTWARE DESCRIPTION
Open Software Description (OSD) è un formato di dati utilizzato per consentire l'aggiornamento e l'installazione di software tramite Internet. Questo formato è particolarmente utile per notificare agli utenti la disponibilità di nuove versioni di software e per fornire un meccanismo per ottenere i programmi da Internet.

ELECTRONIC DATA INTERCHANGE
Electronic Data Interchange (EDI) viene utilizzato correntemente in tutto il mondo per lo scambio di dati e per il supporto delle transazioni. Nell'implementazione corrente tuttavia può essere utilizzato solo da organizzazioni che hanno impostato lo scambio di informazioni mediante sistemi compatibili. Il linguaggio XML può ampliare la portata di EDI e renderlo più accessibile a un maggior numero di organizzazioni.

3.13 SPAZI DEI NOMI XML (XML NAMESPACE)

Come sappiamo XML è un linguaggio per definire linguaggi, cioè insiemi di marcatori personalizzati e le loro sintassi di utilizzo. Questa operazione avviene attraverso i DTD che definiscono gli insiemi di marcatori che saranno utilizzati nei documenti e le loro regole di nidificazione. I marcatori XML che saranno definiti nei DTD adottati sui siti Web possono anche non essere inventati dai Webmaster, ma possono appartenere a repertori standard per i vari domini applicativi (i cosiddetti XML *namespace*), garantendo l'uniformità sintattica delle pagine Web necessaria al funzionamento degli agenti.
I Namespace permettono la creazione e l'uso di marcatori ambigui, ovvero con lo stesso nome, ma in riferimento a significati e ambienti diversi utilizzando costrutti con nomi non equivoci. Pensiamo per esempio a documenti di una rivista in cui la parola "titolo" a secondo del contesto può referenziare il titolo di una rivista, ma anche il ruolo di un giornalista all'interno della struttura aziendale.
I dati, ovvero il contenuto di un documento XML, vengono recuperati analizzando i singoli nodi all'interno del documento, meccanismo consentito dal fatto che la struttura gerarchica dei documenti XML e le regole di validità e di ben formato che gestiscono la creazione dei documenti XML garantiscono che ogni nodo presente in un documento è unico. Questo garantisce a sua volta che esista un solo riferimento per ciascun nodo. L'utilizzo di documenti XML in un ambiente di collaborazione potrebbe tuttavia dare luogo a potenziali problemi. Ad esempio, due o più documenti potrebbero contenere elementi con gli stessi nomi, ma con semantica differente. I documenti possono essere strutturati nello stesso modo. Qualora fosse necessario utilizzare entrambi i documenti in un unico ambiente, la sovrapposizione di elementi sarebbe causa di confusione. Consideriamo ad esempio:

```
<AUTOMOBILE>
 <ID>232-HDF</ID>
</AUTOMOBILE>

<DOG>
 <ID>Rover</ID>
</DOG>
```

Gli elementi Automobile e Dog contengono un elemento Id ciascuno, ma tale Id assume significato differente nei due casi. Se questi elementi provenienti da fonti diverse sono stati combinati in un solo documento, gli elementi Id perdono il significato originale.
Questo problema, tutt'altro che trascurabile, potrebbe aggravarsi parallelemente alla diffusione dell'utilizzo del linguaggio XML sul Web e nelle organizzazioni. La soluzione è offerta dagli spazi dei nomi, che consentono di creare nomi univoci indipendentemente dalla posizione in cui gli elementi vengono utilizzati, garantendo l'uniformità sintattica delle pagine Web.

3.13.1 CREAZIONE DI NOMI UNIVOCI TRAMITE GLI SPAZI DEI NOMI XML

La definizione *spazio dei nomi* è utilizzata dal programmatore tradizionale per indicare un gruppo di nomi in cui non esistono duplicati. Poiché la natura del linguaggio XML consente di definire set di tag personalizzati, che potrebbero dar luogo a nomi duplicati nei documenti XML, nel linguaggio XML gli spazi dei nomi offrono caratteristiche aggiuntive. Costituiscono infatti una metodologia per la creazione di nomi universalmente univoci in un documento XML identificando i nomi degli elementi con una risorsa esterna univoca. Nel linguaggio XML uno *spazio dei nomi* è pertanto una raccolta di nomi identificata da un URI e può essere qualificato o non qualificato.

NOMI QUALIFICATI
Nell'XML un nome qualificato si compone in due parti*: il nome dello spazio dei nomi* e la *parte locale*. Il nome dello spazio dei nomi, ovvero un URI, definisce lo spazio dei nomi, mentre la parte locale corrisponde al nome dell'elemento o dell'attributo del documento locale. Poiché l'URI è sempre univoco, il nome dello spazio dei nomi crea insieme alla parte locale un nome di elemento universalmente univoco. Per poter utilizzare uno spazio dei nomi in documento XML, è necessario includere una *dichiarazione dello spazio dei nomi* nel prologo del documento. E' inoltre possibile includere nella dichiarazione un *prefisso dello spazio dei nomi*. Utilizzando i due punti (:), il prefisso può essere aggiunto alla parte locale in modo da associarla al nome dello spazio dei nomi. Nel documento riportato nell'esempio che segue, i due spazi dei nomi vengono dichiarati con prefissi, quindi utilizzati nel documento.

```
<?xml version="1.0"?>
<?xml:namespace ns=http://inventory/schema/ns prefix="inv"?>
<?xml:namespace ns=http://wildflowers/schema/ns prefix="wf"?>
<PRODUCT>
 <PNAME>Test1</PNAME>
 <inv:quantity>1</inv:quantity>
 <wf:price>323</wf:price>
 <DATE>6/1</DATE>
</PRODUCT>
```

In questo esempio di codice, i prefissi vengono utilizzati per identificare elementi appartenenti allo spazio dei nomi selezionato. Si otterranno in questo modo nomi univoci, ma anche la conservazione del valore semantico dei nomi. Gli elementi inv:quantity e wf:price contengono nomi completamente qualificati che risulteranno univoci indipendemente dalla posizione in cui sono stati utilizzati. Il prefisso è parte del nome dell'elemento e deve essere sempre incluso in modo tale da indicare che l'elemento appartiene allo spazio dei nomi.

NOMI NON QUALIFICATI
Un nome non qualificato non dispone di nome associato al nome dello spazio dei nomi. I nomi di elementi XML tipici non sono qualificati poiché non specificano uno spazio dei nomi.

AREA DI VALIDITA' DELLO SPAZIO DEI NOMI
Il prologo non è l'unica opzione disponibile per la posizione della dichiarazione dello spazio dei nomi. E' infatti possibile includere tale dichiarazione direttamente all'interno di un elemento appartenente allo spazio dei nomi. A questo scopo, è sufficiente includere la

dichiarazione la prima volta che si utilizza l'elemento, come mostrato nell'esempio che segue:

```
<PRODUCT>
 <PNAME>Test1</PNAME>
 <inv:quantity>1</inv:quantity>
 <wf:price xmlns:wf="urn:shemas-wildflowers-com:xml-prices">
  323
 </wf:price>
 <DATE>6/1</DATE>
</PRODUCT>
```

Lo spazio dei nomi risulta quindi disponibile nel contesto dell'elemento specifico, ovvero tale elemento e tutti i relativi elementi secondari possono utilizzare lo spazio dei nomi. Se dichiarato nell'elemento del documneto, lo spazio dei nomi può essere utilizzato nell'intero documento.

SPAZIO DEI NOMI PREDEFININTI

E' possibile impostare come predefinito uno spazio dei nomi dichiarandolo senza l'assegnazione di un prefisso. In questo caso, lo spazio dei nomi viene considerato all'interno del contesto dell'elemento in cui è stato dichiarato, come mostrato nel seguente esempio di codice:

```
<CATALOG>
 <INDEX>
  <ITEM>Trees</ITEM>
  <ITEM>Wildflowers</ITEM>
  §
 </INDEX>

 <PRODUCT xmlns:wf="urn:shemas-wildflowers-com>
  <NAME>Bloodroot</NAME>
  <QUANTITY>10</QUNTITY>
  <PRICE>$2.44</PRICE>
 </PRODUCT>
</CATALOG>
```

L'elemento Product contiene la dichiarazione di uno spazio dei nomi senza prefisso associato. In quanto tale, lo spazio dei nomi viene utilizzato per l'elemento Product e tutti i relativi elementi secondari, ma per nessun altro elemento oltre Product.

DICHIARAZIONE DELLO SPAZIO DEI NOMI COME URL O URN

In quanto univoci, gli URL possono essere utilizzati per rendere univoci i nomi degli spazi dei nomi. Se uno spazio dei nomi è mappato a un URL, tale spazio dei nomi risulterà univoco nell'intero contesto in cui viene utilizzato.
Un'altra situazione tipica è rappresentata dall'esistenza di uno schema dello spazio dei nomi che identifica tutti i nomi contenuti nello spazio dei nomi e il modo in cui sono strutturati. Il nome dello spazio dei nomi XML non fornisce alcun meccanismo per il recupero di tale schema, meccanismo che tuttavia può essere reso disponibile utilizzzando gli URN (Uniform Resource Names).
Un URN consente di individuare e recuperare un file di schema che definisce un determinato spazio dei nomi. Sebbene funzionalità simili possano essere fornite da un

comune URL, a questo scopo l'URN è più efficiente e facile da gestire poiché può essere riferito a più URL.
Il codice seguente mostra l'utilizzo di un URN nel contesto di uno spazio dei nomi XML:

```
<CATALOG>
 <INDEX>
  <ITEM>Trees</ITEM>
  <ITEM>Wildflowers</ITEM>
  §
 </INDEX>

 <wf:product xmlns:wf="urn:shemas-wildflowers-com>
  <wf:name>Bloodroot>/wf:name>
  <QUANTITY>10</QUNTITY>
  <PRICE>$2.44</PRICE>
 </wf:product>
</CATALOG>
```

Lo schema relativo allo spazio dei nomi è disponibile nella posizione identificata dall'URN e l'applicazione di elaborazione è in grado di recuperare tale schema. Lo schema contiene informazioni dettagliate relative agli elementi dello spazio dei nomi utilizzabili nel documento.

SPAZIO DEI NOMI DEGLI ATTRIBUTI
Gli spazi dei nomi sono applicabili così come agli elementi, anche agli attributi. Ad esempio:

```
<wf:product TYPE="plant" class:kingdom="plantae"
 xmlns:wf="urn:wildflowers:schemas:product"
 xmlns:class="urn:bio:botany:classification">
 <PNAME>Test1</PNAME>
 <QUANTITY>1</QUNTITY>
 <PRICE>323</PRICE>
 <DATE>6/1</DATE>
</wf:product>
```

In questo esempio sia all'elemento wf:product che all'attributo *class:kingdom* sono associate dichiarazioni degli spazi dei nomi. Lo spazio dei nomi degli attributi viene utilizzato in modo simile allo spazio dei nomi degli elementi.
L'importanza degli spazi dei nomi aumenta parallelemente allo sviluppo di nuovi vocabolari e di nuove tecnologie basate sul linguaggio XML. Sono comunque già tecnologie che utilizzano gli spazi dei nomi, tra cui XML-Data, i tipi di dati XML e il linguaggio SMIL (Syncronized Multimedia Integration Language).

APPENDICE. ESEMPIO DI FILE XML E RELATIVA DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE radice SYSTEM "Curriculum.dtd">
<radice>
 <curriculum>
  <dati>
   <nome>Luca</nome>
   <cognome>Rossi</cognome>
   <sesso>Maschile</sesso>
   <indirizzo>
    <via>Via Degli Angeli 35</via>
    <cap>21057</cap>
    <city>Olgiate Olona</city>
    <provincia>VA</provincia>
    <nazione>Italia</nazione>
   </indirizzo>
   <telefono>03313434322</telefono>
   <posta>rossi@mail.com</posta>
  </dati>
  <studi>
   <maturita>
    <titolo>Maturita' Scientifica</titolo>
    <voto>48/60</voto>
   </maturita>
   <laurea>
    <universita>Statale di Milano</universita>
    <facolta>Informatica</facolta>
    <votol>110</votol>
   </laurea>
  </studi>
  <esperienze>Da 3 anni responsabile tecnico
"Inforscuola"</esperienze>
  <lingua>Inglese</lingua>
 </curriculum>

 <curriculum>
  <dati>
   <nome>Marco</nome>
   <cognome>Colombo</cognome>
   <sesso>Maschile</sesso>
   <indirizzo>
    <via>Via De Amicis 3</via>
    <cap>21057</cap>
    <city>MILANO</city>
    <provincia>VA</provincia>
    <nazione>Italia</nazione>
   </indirizzo>
   <telefono>0231534322</telefono>
```

```
  <posta>colombo@hotmail.com</posta>
 </dati>
 <studi>
  <maturita>
   <titolo>Maturita' Classica</titolo>
   <voto>54/60</voto>
  </maturita>
  <laurea>
   <universita>Statale di Milano</universita>
   <facolta>Filosofia</facolta>
   <votol>90</votol>
  </laurea>
 </studi>
 <esperienze>Da 2 anni collabora con l'ufficio
  Arte e cultura del Corriere</esperienze>
 <lingua>Spagnolo</lingua>
</curriculum>

<curriculum>
 <dati>
  <nome>Francesca</nome>
  <cognome>Marillo</cognome>
  <sesso>femminile</sesso>
  <indirizzo>
   <via>Via Piave 7</via>
   <cap>21057</cap>
   <city>Como</city>
   <provincia>CO</provincia>
   <nazione>Italia</nazione>
  </indirizzo>
  <telefono>031634340</telefono>
  <posta>marillo@tin.it</posta>
 </dati>
 <studi>
  <maturita>
   <titolo>Maturita' Scientifica</titolo>
   <voto>38/60</voto>
  </maturita>
  <laurea>
   <universita>Cattolica</universita>
   <facolta>Scienze Politiche</facolta>
   <votol>106</votol>
  </laurea>
 </studi>
 <esperienze>Nessuna</esperienze>
 <lingua>Tedesco</lingua>
</curriculum>

<curriculum>
 <dati>
  <nome>Stefano</nome>
  <cognome>Bassi</cognome>
```

```
   <sesso>maschile</sesso>
   <indirizzo>
    <via>Via Piave 7</via>
    <cap>21057</cap>
    <city>Como</city>
    <provincia>CO</provincia>
    <nazione>Italia</nazione>
   </indirizzo>
   <telefono>031908765</telefono>
   <posta>stefano@cio.it</posta>
  </dati>
  <studi>
   <maturita>
    <titolo>Maturita' Tecnica Industriale in
meccanica</titolo>
    <voto>42/60</voto>
   </maturita>
   <laurea>
    <universita>Politecnico</universita>
    <facolta>Ingegneria Civile</facolta>
    <votol>106</votol>
   </laurea>
  </studi>
  <esperienze>Impiegato al comune di
Como</esperienze>
  <lingua>Tedesco</lingua>
 </curriculum>

 <curriculum>
  <dati>
   <nome>Paolo</nome>
   <cognome>Bellucci</cognome>
   <sesso>maschile</sesso>
   <indirizzo>
    <via>Via Piave 7</via>
    <cap>21057</cap>
    <city>Seregno</city>
    <provincia>MI</provincia>
    <nazione>Italia</nazione>
   </indirizzo>
   <telefono>0234546787</telefono>
   <posta>bellucci@dido.it</posta>
  </dati>
  <studi>
   <maturita>
    <titolo>Maturita' Artistica</titolo>
    <voto>60/60</voto>
   </maturita>
   <laurea>
    <universita>Politecnico</universita>
    <facolta>Architettura</facolta>
    <votol>101</votol>
```

```
   </laurea>
  </studi>
  <esperienze>Nessuna</esperienze>
  <lingua>Tedesco</lingua>
 </curriculum>

 <curriculum>
  <dati>
   <nome>Camilla</nome>
   <cognome>Silvestri</cognome>
   <sesso>femminile</sesso>
   <indirizzo>
    <via>Via Piave 7</via>
    <cap>21057</cap>
    <city>Torino</city>
    <provincia>TO</provincia>
    <nazione>Italia</nazione>
   </indirizzo>
   <telefono>03434322098</telefono>
   <posta>camilla@tin.it</posta>
  </dati>
  <studi>
   <maturita>
    <titolo>Maturita' Scientifica</titolo>
    <voto>46/60</voto>
   </maturita>
   <laurea>
    <universita>Cattolica</universita>
    <facolta>Scienze Ambientali</facolta>
    <votol>106</votol>
   </laurea>
  </studi>
  <esperienze>Lavora part-time presso uno studio
veterinario</esperienze>
  <lingua>Inglese</lingua>
 </curriculum>
</radice>
```

DTD:

```
<!ELEMENT radice (curriculum+)>
<!ELEMENT curriculum (dati, studi, esperienze, lingua,  hobbies*)>
<!ELEMENT dati (nome+, cognome, sesso, indirizzo, telefono*,
posta*)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT sesso (#PCDATA)>
<!ELEMENT indirizzo (via, cap, city, provincia, nazione)>
<!ELEMENT via (#PCDATA)>
<!ELEMENT cap (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT provincia (#PCDATA)>
<!ELEMENT nazione (#PCDATA)>
<!ELEMENT telefono (#PCDATA)>
<!ELEMENT posta (#PCDATA)>
<!ELEMENT studi (maturita*, laurea*)>
<!ELEMENT maturita (titolo, voto)>
<!ELEMENT laurea (universita, facolta, votol)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT voto (#PCDATA)>
<!ELEMENT universita (#PCDATA)>
<!ELEMENT facolta (#PCDATA)>
<!ELEMENT votol (#PCDATA)>
<!ELEMENT esperienze (#PCDATA)>
<!ELEMENT lingua (#PCDATA)>
```