

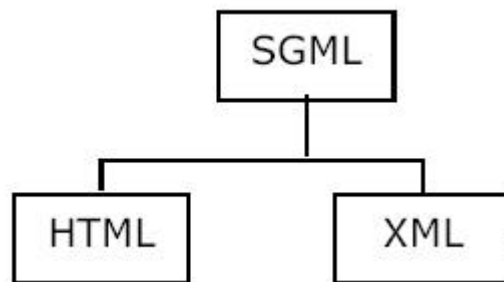
Indice

8. XML – EXTENSIBLE MARKUP LANGUAGE	2
8.1 USI DI XML	3
8.2 CARATTERISTICHE DI XML.....	4
8.3 ELEMENTI	7
8.4 ATTRIBUTI	8
8.5 INFORMAZIONI DI CONTROLLO	8
8.5.1 <i>Commenti</i>	9
8.5.2 <i>Istruzioni di elaborazione</i>	9
8.6 DOCUMENT TYPE DEFINITION (DTD)	10
8.6.1 <i>Elementi</i>	11
8.6.2 <i>Attributi</i>	13
8.6.3 <i>Documenti XML validi</i>	14
8.7 ENTITÀ.....	16
8.8 SPAZI DEI NOMI (NAMESPACES).....	17
8.9 VISUALIZZAZIONE DI DOCUMENTI XML TRAMITE FOGLI DI STILE.....	19

8. XML – Extensible Markup Language

XML è un linguaggio di marcatura estensibile (insieme di regole per creare altri linguaggi di marcatura) sviluppato e controllato dal World Wide Web Consortium (W3C).

Il suo sviluppo è iniziato nel 1996 ed è stato adottato come standard da W3C nel 1998. L'attuale versione di questo standard è la 1.0, ma è in progressiva evoluzione. XML è un sottoinsieme di SGML (Standard Generalized Markup Language) che definisce regole per scrivere linguaggi di marcatura, e non comprende tutte le funzionalità di SGML.



XSL sta per Extensible Stylesheet Language e indica i fogli di stile creati specificatamente per documenti XML

La marcatura consiste in un insieme di istruzioni, in genere rappresentate da tag, che si aggiungono ad un file. I tag sono utilizzati da un'applicazione in grado di interpretarli, per controllare la struttura del file o la presentazione delle informazioni contenute nel file marcato.

XML, contrariamente a HTML, è estensibile (metalinguaggio), non avendo tag predefiniti, e consente di scrivere altri linguaggi. Diverse organizzazioni hanno creato differenti “vocabolari” di marcatura specializzati in determinati settori. La principale differenza tra HTML e XML risiede nel fatto che il primo, descrivendo le caratteristiche di presentazione delle informazioni, è uno standard per la presentazione, mentre il secondo, descrivendo il contenuto delle informazioni, è uno standard per l'elaborazione dei dati. XML, quindi, è usato per descrivere la struttura di un documento e non il modo con il quale deve essere presentato. Un documento facilmente interpretabile deve contenere almeno tre parti fondamentali rappresentate in modo distinto: contenuto, struttura, stile (rappresentazione “tipografica”).

In altri termini, HTML è un metodo universale per visualizzare i dati, XML è un metodo universale per “lavorare direttamente” con i dati, per descrivere e scambiare informazioni strutturate. I linguaggi espressi tramite la grammatica XML, inoltre, sono conformi ad uno standard internazionale, possono essere “compresi” da diverse applicazioni e, al tempo stesso, sono calibrati per soddisfare specifiche esigenze proprio perché creati dall'utente. L'interoperabilità tra applicazioni può essere facilitata dall'uso massivo dello standard XML sfruttando le caratteristiche

di essere indipendente da sistemi e piattaforme. Tuttavia, non contenendo tag predefiniti ed essendo estensibile, sorge la necessità di condividere con altri attori (uomini o sistemi) il linguaggio creato, dichiarando il significato delle estensioni usate. La descrizione della struttura di un file XML è fatta tramite DTD (document type definition, definizione del tipo di documento).

Alcune esigenze che spingono verso una tecnologia più evoluta rispetto ad HTML possono essere sintetizzate:

- ? internazionalizzazione del publishing elettronico che richiede indipendenza dalle piattaforme, dai media, dai linguaggi usati: il medesimo contenuto deve essere utilizzabile “ovunque”.
- ? nuove applicazioni web based che richiedono scambio di informazioni tra database, manipolazione dei dati da parte delle postazioni di lavoro (selezione, ordinamento,...), personalizzazione delle informazioni con agenti “intelligenti”, ecc.

La tecnica degli script, già usata per realizzare alcune delle funzionalità indicate, comporta difficoltà nella gestione (creazione, manutenzione, aggiornamento), e non consente di governare agevolmente la crescita delle informazioni da trattare. Alcuni obiettivi ufficiali dello standard XML, definiti dal W3C, sono:

- ? facilità e usabilità su Internet
- ? supporto di ampie gamme di applicazioni
- ? facilità nella scrittura di programmi per l’elaborazione di documenti XML
- ? leggibilità da parte delle persone e ragionevole chiarezza
- ? rapidità della fase di disegno
- ? disegno formale e conciso

8.1 Usi di XML

XML si presta a poter essere utilizzato in un’ampia varietà di settori, particolarmente ove risulta critica l’identificazione del contenuto informativo. L’uso più comune di XML è dato dall’elaborazione di testi, essenzialmente in quanto XML fornisce un insieme di regole specifiche per la creazione e la strutturazione di documenti. In quanto tale, XML può definirsi complementare piuttosto che concorrente rispetto ad HTML poiché quest’ultimo fornisce istruzioni di formattazione e visualizzazione unitamente al contenuto informativo XML.

Un’altra area è costituita dai database, in particolare per il publishing su Web. Si adatta bene per applicazioni che mostrano lo stesso documento in modi diversi con diversi dispositivi di visualizzazione.

Una terza area è data dai motori di ricerca. I motori di ricerca che fanno uso di HTML possono fornire risultati poco accurati, anche se eseguite con parametri di ricerca adeguati.

Dal momento che XML tratta il significato e la struttura dei dati è in grado di migliorare in maniera significativa i risultati delle ricerche. L'area di utilizzo che presenta maggiori prospettive di crescita è lo sviluppo di applicazioni. A causa delle caratteristiche mirate all'interscambio di dati, XML è una tecnologia adatta per applicazioni di tipo distribuito, in particolar modo nei casi in cui i dati sono soggetti a frequenti modifiche.

Fra le prime organizzazioni che hanno adottato XML figurano gli operatori di e-commerce. In particolare, le aree applicative che possono trarre beneficio dall'uso di XML sono Business to Business (B2B) e Application to Application (A2A).

8.2 Caratteristiche di XML

Struttura: possono essere rappresentati dati gerarchici complessi

Validazione: i dati possono essere controllati nella loro "correttezza strutturale"

Indipendenza dai media: lo stesso contenuto può essere pubblicato su più media

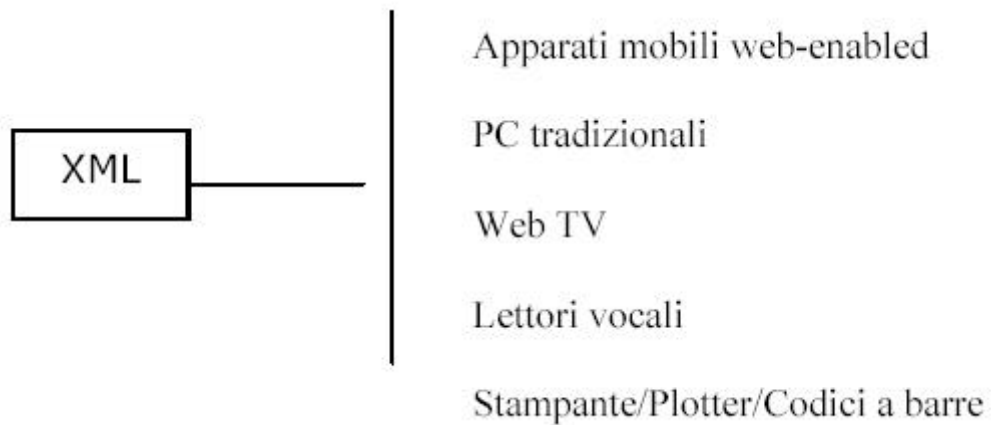
Svantaggi

- 1) performance
- 2) nuovo e in continua evoluzione
- 3) sicurezza

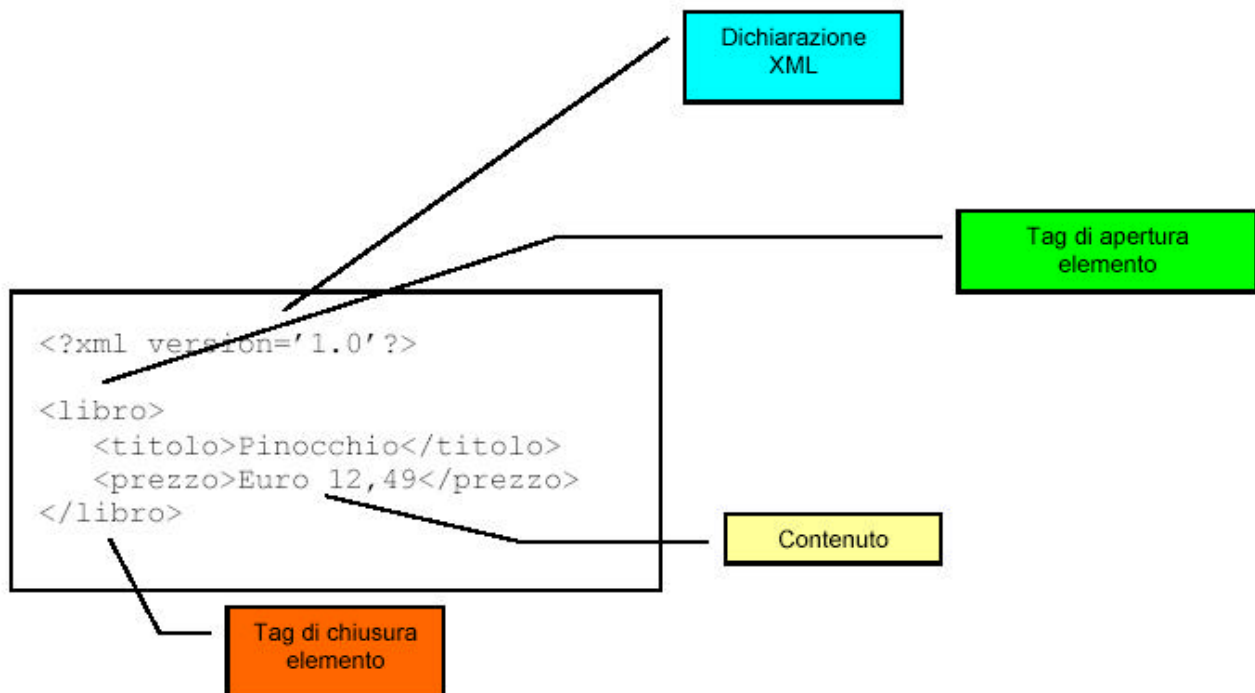
Tecnologia post-html

Gli script sono usati, in genere, per scopi specifici, la loro gestione può risultare difficile, non facilitano la gestione e "crescita" dei dati.

Interoperabilità generalizzata



Dal punto di vista sintattico l'elemento è composto dal tag di apertura, dal tag di chiusura e dal contenuto:



Esempio

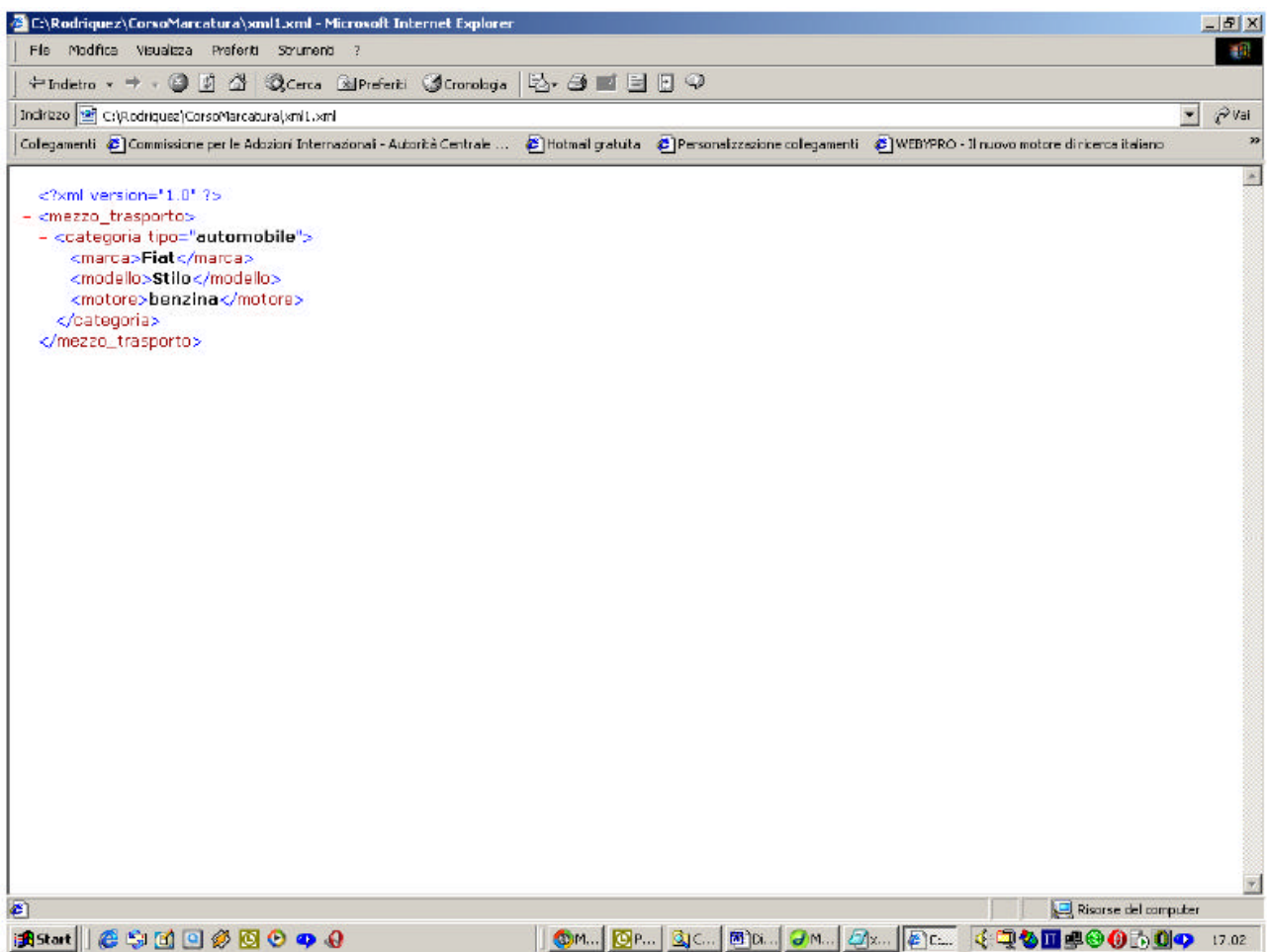
```
<?xml version="1.0"?>  
  <mezzo_trasporto>  
    <categoria tipo="automobile">  
      <marca>Fiat</marca>  
      <modello>Stilo</modello>  
      <motore>benzina</motore>  
    </categoria>
```

```
</mezzo_trasporto>
```

Si noti che:

- o i tag sono autoesplicativi e indicano la natura del dato
- o non sono presenti tag di gestione della formattazione

Aprendo il file con un browser che include un parser XML si ottiene l'immagine seguente:



Un documento XML può essere:

- o ben formato
- o valido

Un documento XML ben formato significa sintatticamente corretto (ciò può essere verificato dal

parser). E' valido se, *oltre* ad essere ben formato, è conforme alla DTD interna o esterna utilizzata (esistono anche parser che eseguono il controllo di validità).

Si tenga presente che una DTD non è un documento XML.

Esempio documento ben formato:

```
<?xml version="1.0"?>
  <saluto>Hello, world</saluto>
```

Esempio documento non ben formato:

```
<?xml version="1.0"?>
  <saluto>Hello, world</Saluto]
```

(il parser XML genera un errore perché il tag iniziale non corrisponde esattamente al tag finale: la grammatica XML è sensibile ai caratteri maiuscoli/miniscoli)

Es. documento valido

```
<!DOCTYPE saluto SYSTEM "saluto.dtd">
  <!ELEMENT saluto (#PCDATA)>
```

il documento è valido perché è conforme a questa DTD (successivamente verranno esaminate le caratteristiche della DTD)

Un documento XML è composto da:

- elementi
- informazioni di controllo
- entità

8.3 Elementi

Le unità di informazioni marcate con specifici tag sono dette elementi.

Il documento deve avere *un solo* elemento radice e un numero variabile di sezioni. Anche le sezioni sono elementi e sono delimitate da tag.

I tag dei vari elementi devono essere nidificati (significa che i tag di chiusura devono apparire in ordine inverso rispetto all'ordine con il quale appaiono i rispettivi tag di apertura) e sono obbligatori i tag di chiusura.

Inoltre la grammatica di XML è case-sensitive.

8.4 Attributi

Ogni elemento può essere integrato con ulteriori informazioni. Tali informazioni associate agli elementi sono dette attributi, e sono usati per contenere valori che descrivono “meglio” gli elementi ai quali si riferiscono. E’ da sottolineare che la definizione degli elementi e dei loro attributi descrittivi è un passo importante. Decidere se un attributo deve essere portato a livello di elemento così come selezionare i possibili attributi che meglio descrivono gli elementi è un compito che rientra nella fasi di analisi e modellazione dei dati.

In maniera indicativa si potrebbe affermare che i dati essenziali del modello (ovvero quei dati da sottoporre ad elaborazione diretta) dovrebbero essere dichiarati come elementi. Gli attributi dovrebbero invece essere quei dati che rivestono significato solo nel contesto di riferimento o che non variano (significativamente) con il trascorrere del tempo.

Ad es. in un inventario di magazzino si suppone che il nome di un bene non vari nel tempo, mentre la quantità ed il valore possono variare. Di conseguenza il nome potrebbe essere un attributo e la quantità ed il valore dovrebbero essere codificati come elementi.

Un altro esempio può essere costituito dalla necessità di trattare informazioni di minore importanza che si desidera memorizzare ma non necessariamente visualizzare e, pertanto, tali informazioni possono essere assegnate ad attributi invece di collocarle nel contenuto di un documento (la visualizzazione di un documento XML con fogli di stile CSS non mostra gli attributi).

Inoltre si consideri che:

- ? gli attributi non sono facilmente espandibili per apportare successivamente eventuali modifiche
- ? è più complesso manipolare gli attributi tramite il codice del programma
- ? è più complesso testare i valori degli attributi con una DTD

8.5 Informazioni di controllo

Le informazioni di controllo sono:

- ? commenti
- ? istruzioni di elaborazione
- ? Document Type Definition

8.5.1 Commenti

Lo scopo dei commenti è quello di facilitare le operazioni di sviluppo e manutenzione, anche a distanza di tempo, fornendo chiarimenti, esempi, presupposti, regole, ecc., soprattutto quando il lavoro è svolto da più persone che lavorano in gruppo o quando si tratta di DTD *pubbliche*. La sintassi per l'inserimento di commenti è la stessa sia per i documenti XML sia per i DTD.

```
<!--Questo è un esempio di commento XML -->
```

I commenti possono essere inseriti in qualsiasi punto del documento XML purchè esterno ad altri tag e non possono contenere i caratteri “--“ (doppio trattino).

Ad es. il seguente commento non è valido perché collocato all'interno di un tag:

```
<elemento <!-- Questo è un esempio di commento XML non valido --> >
Documento XML </elemento>
```

I commenti possono essere inseriti anche all'interno della DTD.

8.5.2 Istruzioni di elaborazione

Lo scopo delle istruzioni di elaborazione è quello di “comunicare” con le applicazioni, fornendo informazioni che il documento XML passa all'applicazione.

Esempi di istruzioni:

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

Questa istruzione comunica all'applicazione che il contenuto del file è conforme agli standard XML versione 1.0. Deve necessariamente essere la prima riga del file XML. Sebbene lo standard preveda l'obbligatorietà di questa istruzione alcuni parser (es.: Explorer), in mancanza, la considerano implicitamente presente.

```
<?xml-stylesheet type="text/css" href="Biblioteca01.css"?>
```

Questa istruzione comunica all'applicazione di utilizzare il foglio di stile disponibile nel file Biblioteca01.css.

Analogamente a quanto previsto per i commenti, le istruzioni di elaborazione possono essere inserite in qualsiasi punto del documento XML purchè esterno ad altri tag.

8.6 Document Type Definition (DTD)

L'utilizzo delle DTD è opzionale in quanto in determinati casi non sono rilevanti mentre in altre situazioni rivestono un aspetto determinante. Non sono rilevanti, ad es., per informazioni scambiate all'interno di applicazioni, come avviene in un unico database, o di una singola organizzazione, ma sono critiche per scambi informativi tra applicazioni e tra organizzazioni diverse.

Ciascun documento XML può essere associato ad una DTD che può essere contenuta nello stesso file XML che la utilizza o può essere specificata in un file esterno. In quest'ultimo caso la DTD potrà essere utilizzata da più file XML. La DTD contiene la particolare grammatica specificata per una determinata struttura XML. Contiene quindi l'indicazione degli elementi, il loro ordine di apparizione, i tag utilizzati. Tale grammatica verrà usata dai parser validatori per determinare, oltre alla correttezza sintattica del file (ben formato), anche che è conforme alla grammatica stabilita (valido).

```
<!DOCTYPE Ricetta SYSTEM "Ricetta.dtd">
```

Questa dichiarazione, inserita nel file XML, comunica al parser che il file usa una DTD specificata in un altro file denominato Ricetta.dtd e che tale DTD si trova nella stessa directory del file XML. Per convenzione si usa per la DTD lo stesso nome del file XML (Ricetta.xml - Ricetta.dtd).

La parola chiave SYSTEM è usata per indicare una DTD sviluppata dall'utente per sue proprie esigenze e che ha in genere validità solo all'interno della propria organizzazione.

La parola chiave PUBLIC indica invece una DTD sviluppata e riconosciuta pubblicamente, anche a livello internazionale.

Possono essere elaborate sia dichiarazioni interne sia dichiarazioni esterne. Quelle interne *prevalgono* su quelle esterne.

Come accennato, la DTD contiene le regole di definizione dei tag (markup) e indica gli elementi e il loro ordine nel documento XML e stabilisce, tra l'altro, condizioni obbligatorie e facoltative alle quali i file XML devono conformarsi.

Con la DTD si esprimono quindi i metadati mentre i dati sono contenuti nel file XML.

La strutturazione della DTD riveste un'importanza fondamentale e richiede un'attenta attività di analisi e di modellazione dei dati relativi al contesto di riferimento.

Passi per la definizione di una DTD

- 1) definizione del proprio linguaggio di markup (tag)
- 2) definizione della struttura del documento
- 3) definizione dei vincoli (constraint) e dei valori predefiniti (default) per gli attributi

Il file XML può contenere al suo interno la DTD. In questo caso tutta la DTD è collocata in un unico tag DOCTYPE delimitato da parentesi quadre.

Esempio

```
<?xml version='1.0?'>
<!DOCTYPE recipe [
    <!ELEMENT cookbook (category+)>
    <!--altre istruzioni della DTD
]>
<recipe>
<!--qui si inserisce il resto del file XML
</recipe>
```

8.6.1 Elementi

La struttura di un documento XML è esplicitata nella DTD. Un documento XML è costituito da elementi che possono contenere al loro interno altri elementi. Determinati elementi possono contenere, a loro volta, attributi. Il primo nodo, che contiene tutti gli altri, è detto nodo radice e tutti i documenti XML ne devono contenere obbligatoriamente uno.

Si consideri che gli elementi devono avere un nome univoco nell'ambito di uno stesso documento e che tutti gli elementi sono dichiarati con lo stesso formato.

Il tag element è indicato con un punto esclamativo seguito dalla parola ELEMENT in caratteri maiuscoli, dal nome dato all'elemento e infine da altre informazioni racchiuse tra parentesi. Ogni elemento può contenere altri elementi o può definire dei dati. Se l'elemento contiene altri nodi, nelle parentesi sono indicati i sottoelementi separati da una virgola. Ogni sottoelemento può essere accompagnato da un operatore che indica le regole con le quali il sottoelemento deve essere presente.

```
<!ELEMENT biblioteca (libro+)>
```

Operatori

- ? “*” indica che l'elemento appare più volte o zero volte (es.: immagine*)
- ? “,” suddivide l'elemento in base ad una sequenza di altri elementi nell'esatto ordine con cui la sequenza si presenta (es.: riassunto, corpo del capitolo)
- ? “()” racchiude un gruppo di elementi (es.: item1, item2)
- ? “[|]” indica elementi alternativi, dei quali se ne può presentare soltanto uno (es.: acceso | spento)

- ? “+” indica che l’elemento appare almeno una volta (es.: libro+)
- ? “?” indica che l’elemento appare una sola volta o nessuna (es.: biblioteca?)

Si consideri una semplice rappresentazione di una biblioteca, costituita solo da libri (senza ulteriori dettagli), un catalogo di ricerca , un chiosco multimediale ed eventuali punti di ristorazione. I dati di questi tre elementi devono essere espressi nello stesso ordine. La biblioteca, oltre ai libri, dispone di un solo catalogo di ricerca e, eventualmente, di un solo chiosco di accesso multimediale. Potrebbe infine disporre di almeno due punti di ristorazione.

La dichiarazione del tipo di elemento *biblioteca* (della specifica biblioteca descritta) è:

```
<!ELEMENT biblioteca (libro+, catalogo, chiosco?, puntoristorazione*)>
```

Con questo formalismo è quindi possibile rappresentare in modo rigoroso realtà nelle quali gli elementi si presentano con una data sequenza, in modo opzionale, con ripetizioni, in alternativa. Queste possibilità possono essere espresse anche in modo combinato utilizzando gli operatori e le parentesi, al fine di attribuire le corrette priorità all’interno della combinazione. Si ricorda che ogni documento XML deve contenere un solo elemento radice (elemento di livello più alto all’interno della struttura) che contiene tutte le restanti parti del documento.

Esempio:

```
<!ELEMENT recipe (name, ingredient+, cooking+, serves?, instruction*)>
```

Con questa definizione dell’elemento “recipe” si indica che il “name” appare solo una volta, è obbligatorio che siano presenti almeno un elemento “ingredient” e “cooking”, che è ammesso solo un elemento “serves” e che è previsto un numero non limitato di elementi “instructions”.

PCDATA e CDATA

Gli elementi che contengono dati sono dichiarati con il seguente formato:

```
<!ELEMENT NomeElemento (#PCDATA)>
```

Qui le parentesi contengono il tipo di dati dell’elemento, preceduto dal carattere “#”. I tipi di dato comunemente inclusi negli elementi sono PCDATA e CDATA.

Tutto il testo contenuto all’interno di un documento XML è analizzato dal parser, ad eccezione della parte di testo inclusa in una sezione CDATA.

- ? **CDATA** consiste in un set di caratteri che non viene esaminato dal parser XML. Questo tipo di dato è utile se, ad esempio, il set contiene caratteri usati per la marcatura come &, <, > e non si vuole che il parser li analizzi per evitare la generazione di falsi errori, oppure se la

sezione CDATA contiene parti di script per l'elaborazione dei dati presenti nel documento XML. Un esempio di quest'ultimo caso può essere dato da un documento XML che contiene denominazione, quantità e prezzo unitario di un articolo da acquistare e la sezione CDATA che contiene le istruzioni di elaborazione per il calcolo dell'importo totale dell'articolo acquistato e per la sua visualizzazione.

- ? **PCDATA** (Parsed character data) è invece un set di caratteri analizzato dal parser XML ed è il tipo di dato di default degli elementi.

Si noti che sebbene un documento XML può contenere diversi tipi di dato è preferibile che quelli maggiormente complessi (es.: gif) siano inclusi in entità (per le “entità” vedi paragrafo 8.7).

8.6.2 Attributi

Un elemento XML, oltre a contenere dati o altri elementi, può essere corredato da attributi.

Il concetto di attributo è già presente in HTML, come nell'esempio del tag hyperlynk riportato di seguito:

```
<a href="http://www.governo.it">Link al sito del Governo italiano</a>
```

In questo esempio il tag HTML dell'indirizzo è costituito dal testo sul quale cliccare per raggiungere l'indirizzo di destinazione che è invece un attributo del tag. Gli attributi indicano particolari caratteristiche del contenuto degli elementi. In pratica aggiungono informazioni, e hanno significato solo nell'ambito dell'elemento al quale si accompagnano. Sono composti da una coppia formata dal nome dell'attributo e dal suo valore. Una volta individuati gli elementi, i loro attributi descrittivi si definiscono nella DTD che rappresenta la struttura del documento XML.

La definizione si esegue con la parola chiave ATTLIST, che può includere:

- ? nome attributo dell'elemento
- ? tipo di dato di ogni attributo
- ? eventuale valore di default per ogni attributo
- ? informazioni di controllo

Gli attributi devono essere inseriti dopo il tag di apertura e il nome dell'elemento al quale si riferiscono. All'interno di uno stesso elemento i nomi degli attributi devono essere univoci. Il valore dell'attributo è racchiuso tra virgolette.

Esempio

```
<!ATTLIST tartufo
    quantità CDATA #REQUIRED
    unità CDATA "grammi">
```

Questa dichiarazione definisce gli attributi, “quantità” e “unità” dell’elemento “tartufo”, entrambi di tipo CDATA, il che indica che il testo contenuto nei due attributi non verrà preso in considerazione dal parser XML.

La parola chiave REQUIRED indica che l’attributo quantità è obbligatorio. Ciò significa che se tale attributo non verrà valorizzato quando si usa l’elemento, il parser XML segnalerà l’errore. Si noti che tale errore è segnalato perché l’attributo è dichiarato come obbligatorio ma che, una volta valorizzato, il parser non ne analizzerà il contenuto in quanto il tipo di dato è dichiarato come CDATA.

Infine, l’attributo “unità” non è obbligatorio e presenta un valore di default “grammi”. Se questo attributo non verrà valorizzato nel file XML, verrà usato il valore “grammi”.

Oltre che REQUIRED un attributo può essere definito FIXED (opzionale ma se presente è valido il solo valore specificato) o IMPLIED (opzionale senza valore di default).

Il tipo di un attributo definisce l’insieme dei possibili valori.

Alcuni tipi di attributo possono essere:

- ? CDATA in cui è consentito inserire qualsiasi stringa testuale. Si noti che il tipo attributo CDATA è diverso dalla sezione CDATA

Esempio:

```
<!ATTLIST libro
    titolo CDATA #REQUIRED ....
```

- ? elenco di nomi che indica che il valore di un attributo deve essere uno fra quelli compresi nell’elenco.

Esempio:

```
genere (uomo|donna)
```

- ? esistono altre tipologie di attributi ma si è preferito non riportarli in quanto non sono attinenti al corso.

8.6.3 Documenti XML validi

Un documento XML valido risponde ai seguenti requisiti:

- ? è un documento XML ben formato (un documento non formato correttamente non può essere considerato un documento XML)
- ? include una DTD, dichiarazione del tipo di documento, che ne definisce la struttura
- ? è conforme alla DTD

Attraverso l'inclusione di una DTD si può verificare che il documento XML sia conforme a tale DTD. Si consideri che lo standard XML definisce una DTD come la “grammatica di una classe di documenti”.

La creazione di documenti validi garantisce l'uniformità tra gruppi di documenti simili. La creazione di una singola DTD e la sua inclusione in tali documenti garantiscono l'uniformità di tutti i documenti a tale struttura.

Esempi di *modello di contenuto*

```
<!ELEMENT film (protagonista* | regista | produttore)>
```

In base a tale modello di contenuto saranno validi ognuno dei tre elementi seguenti:

1)

```
<film >
    <protagonista>Totò</protagonista>
    <protagonista>Massimo Troisi</protagonista>
</film>
```

2)

```
<film >
    <regista>Steno</regista>
</film>
```

3)

```
<film/> (Elemento vuoto)
```

In un documento XML valido vanno dichiarati anche tutti gli attributi associati agli elementi.

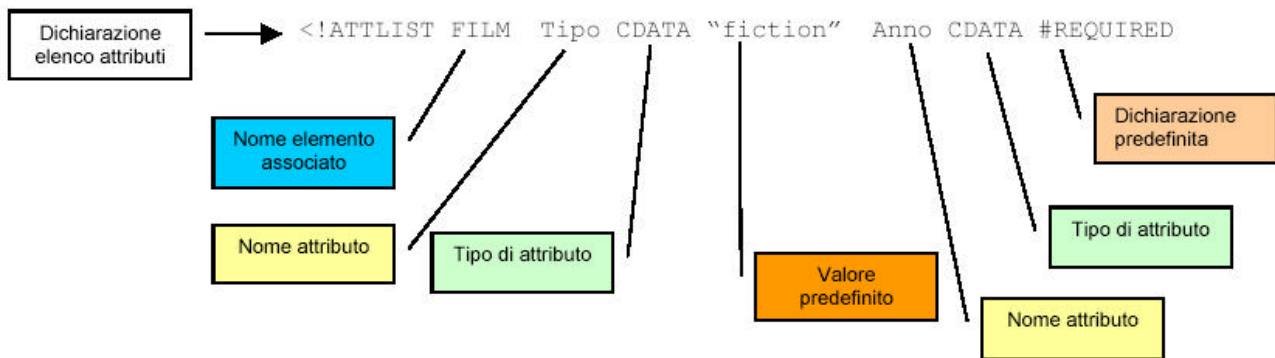
Il corrispondente tipo di tag è detto *dichiarazione dell'elenco degli attributi*.

Esempio

```
<!ELEMENT FILM (TITOLO, (PROTAGONISTA* | REGISTA | PRODUTTORE))>
```

```
<!ATTLIST FILM Tipo CDATA "fiction" Anno CDATA #REQUIRED
```

Con la prima istruzione è stato dichiarato l'elemento denominato film, con la seconda sono stati dichiarati due attributi (tipo e anno) per gli elementi film.



Omettendo l'attributo di un elemento verrà automaticamente assegnato il valore predefinito "fiction" mentre all'attributo anno si può assegnare qualsiasi stringa tra virgolette valida.

Anno è un attributo obbligatorio (parola chiave "REQUIRED") e non ha un valore predefinito.

Il tipo di attributo indica il tipo di valore (di tipo stringa, numerico, ecc.) assegnabile all'attributo all'interno del documento.

8.7 Entità

Un documento XML può contenere una o più entità. Le entità sono *oggetti*, non costituiti da elementi semplici, che sono usati di frequente e che possono essere inclusi in più file XML, anziché essere definite di volta in volta per ogni file XML. Si tratta, in altri termini, di nomi simbolici che verranno usati al posto di altre definizioni.

Le entità ospitano un generico contenuto.

Sono dette interne se incluse all'interno del file XML o esterne se ospitate in un altro file.

Possono inoltre essere di tipo *parsed* o *non parsed*. E' preferibile definire le entità che ospitano contenuti complessi come di tipo *non parsed*.

Le entità sono anche un modo per rappresentare caratteri speciali come &, <, >, "

Ad esempio, non è possibile scrivere

```
<test>if prezzo <150 then</test>
```

in quanto il parser interpreta il carattere < come l'inizio di un nuovo elemento, e genererebbe un errore.

Il testo precedente deve essere scritto come segue:

```
<test>if prezzo &lt;150 then</test>
```

Per l'utilizzo di caratteri speciali si riporta una tabella riepilogativa

Entità	Descrizione	Valore
<	Parentesi angolare sinistra (minore)	<

>	Parentesi angolare destra (maggiore)	>
&	e commerciale (&)	&
'	Apostrofo	'
"	Virgolette doppie	“

Si tenga comunque presente che i caratteri illegali in XML sono propriamente “<” e “&”, mentre è opportuno sostituire gli altri indicati nella tabella, anche se in realtà il loro uso è ammesso.

E’ anche possibile includere nelle entità i caratteri speciali rappresentandoli mediante il formato &#nnn in cui *nnn* indica il codice decimale di un carattere.

Le entità possono essere anche usate per includere nella struttura il contenuto di file esterni.

Entità interne

Le entità interne sono utili per creare contenuti informativi che devono essere ripetuti frequentemente all’interno della struttura. Oltre a facilitare la leggibilità e la manutenzione dello schema, le dimensioni del file XML sorgente saranno ridotte.

```
<!ELEMENT TEATRO “Teatro San Carlo”>
```

Questa istruzione può essere usata come indicato di seguito, ogni volta che occorre la dicitura “Teatro San Carlo”:

```
<item> Il tour si conclude con uno spettacolo al &TEATRO;</item>
```

L’uso di un’entità interna, come illustrato nell’esempio, è indicato con il carattere “&” seguito dal carattere “;”

Entità esterne

```
<!ELEMENT logoaziendale SYSTEM “logo.png”>
```

L’istruzione indica come si crea un’entità esterna (cioè definita in un’altra DTD) che contiene un’immagine, il logo aziendale.

La parola chiave SYSTEM indica, come già visto, che si tratta di una DTD creata dall’utente ad uso interno all’organizzazione.

8.8 Spazi dei nomi (*namespaces*)

Poiché i nomi degli elementi in XML non sono fissi possono verificarsi conflitti se più documenti usano gli stessi nomi per descrivere elementi differenti. Può quindi accadere, nel caso di numerosi documenti XML creati all’interno di grandi organizzazioni, che alcuni nomi di elementi possano

essere duplicati. Unendo documenti che usano lo stesso nome di elementi ma riferiti a definizioni differenti si generano conflitti. Gli spazi dei nomi forniscono un metodo per evitare tali conflitti.

Gli spazi dei nomi sono, in sintesi, una lista di nomi di tag identificati tramite un riferimento ad una risorsa specifica o ad un URI (Uniform Resource Indicator).

Gli spazi dei nomi sono costituiti da due componenti: un prefisso, che indica dove trovare la risorsa che definisce lo spazio dei nomi, ed una componente locale, che è il nome usato nel documento o nell'applicazione. Il parser XML usa i nomi degli elementi per creare una rappresentazione degli item al suo interno.

Esempio

```
<persona>
  <nome>Ferdinando Rodriquez</nome>
</persona>
<ufficio>
  <nome>Informatica</nome>
  <ubicazione>Primo piano stanza 9999</ubicazione>
</ufficio>
```

L'unione dei due documenti sopraindicati genera un conflitto tra i nomi di elementi in quanto entrambi usano un elemento `<nome>` con diverso significato. E' evidente che nel caso di documenti piccoli è possibile rinominare uno dei due elementi evitando ambiguità, ma il problema può presentarsi nel caso di numerosi schemi creati in grandi organizzazioni.

L'uso dei *namespaces* consente di tenere separati i nomi e ne consente l'utilizzo senza errori da parte dell'applicazione.

Facendo precedere il nome dell'elemento da un prefisso seguito da due punti si creano nomi diversi di elementi, a seguito della concatenazione "*prefisso:nome*".

Usando un prefisso si ottiene

```
<a:persona>
  <a:nome>Ferdinando Rodriquez</a:nome>
</a:persona>
<b:ufficio>
  <b:nome>Informatica</b:nome>
  <b:ubicazione>Primo piano stanza 9999</b:ubicazione>
</b:ufficio>
```

ottenendo quindi due diversi tipi dell'elemento `<nome>`.

Esempio

```

<?xml version="1.0"?>
<!DOCTYPE Recipes SYSTEM "...">
<!xml:namespace ns=http://URL/namespaces/personale prefix="funzionari">
<!xml:namespace ns=http://URL/namespaces/strutture prefix="dipartimenti">
<funzionari:persona>
    <funzionari:nome>Ferdinando Rodriquez</funzionari:nome>
</funzionari:persona>
<dipartimenti:ufficio>
    <dipartimenti:nome>Informatica</b:nome>
    <dipartimenti:ubicazione>Primo piano</dipartimenti:ubicazione>
</dipartimenti:ufficio>

```

Invece di fare uso solo dei prefissi nell'esempio precedente è stato aggiunto un *attributo xmlns* per attribuire all'elemento un nome qualificato associato ad un namespace. Quando un namespace è definito nel tag iniziale di un elemento, tutti gli elementi figli con lo stesso prefisso sono associati al medesimo namespace.

8.9 Visualizzazione di documenti XML tramite fogli di stile

Il metodo più semplice di visualizzare documenti XML consiste nell'usare i fogli di stile, la cui sintassi è già stata esaminata in HTML. Il documento XML con il foglio di stile allegato può essere aperto direttamente tramite browser che supporta i fogli di stile CSS. E' anche il metodo più limitato in quanto, ad es., non consente l'accesso agli attributi né l'elaborazione delle informazioni contenute nel documento.

Contrariamente ad HTML, in XML ciascun utente può liberamente definire i propri elementi ed il browser non può conoscere a priori come visualizzarli in maniera corretta. Il foglio di stile CSS contiene le istruzioni per la formattazione degli elementi del documento XML e, collegando il foglio di stile al documento XML, il browser sarà in grado di comprendere come visualizzare correttamente gli elementi.

La separazione fra le istruzioni tipografiche e i dati consente l'utilizzo dello stesso testo marcato con XML per differenti scopi, come la visualizzazione su web o la stampa, mediante l'associazione delle opportune caratteristiche tipografiche contenute nei fogli di stile.

Esempio

```

LIBRO
{display:block;
margin-top:12pt;
font-size:10pt}

```

Di norma, tranne alcune eccezioni, l'impostazione di una proprietà per un dato elemento è propagata a tutti gli elementi secondari. Se, quindi, l'elemento libro ha elementi secondari (capitoli, pagine, ecc.) questi ereditano le proprietà specificate per l'elemento superiore. Di conseguenza è opportuno iniziare la definizione del foglio di stile partendo dagli elementi superiori ed apportare impostazioni aggiuntive passando via via agli elementi nidificati.

Quando il foglio di stile non contiene alcuna proprietà per un determinato elemento, il browser usa un valore predefinito incorporato nel browser stesso o un valore che l'utente ha impostato. Si tenga comunque presente che gli utenti con problemi di vista potrebbero voler visualizzare i caratteri con dimensioni maggiori. In questo caso tali utenti potrebbero decidere di mantenere il controllo sulla formattazione delle pagine, configurando il browser in modo da attribuire priorità alle impostazioni delle proprietà del browser rispetto a quelle presenti nel foglio di stile.

```
<?xml-stylesheet type="text/css" href="Biblioteca01.css"?>
```

Questa istruzione comunica all'applicazione di utilizzare il foglio di stile disponibile nel file Biblioteca01.css.

Esempio

```
<?xml version="1.0"?>
<!-- Nome file: Biblioteca.xml -->
<?xml-stylesheet type="text/css" href="Biblioteca01.css"?>
  <BIBLIOTECA>
    <BOOK>
      <TITLE>Pinocchio</TITLE>
      <AUTHOR>Collodi</AUTHOR>
      <BINDING>tascabile</BINDING>
      <PAGES>150</PAGES>
      <PRICE>EURO 10,00</PRICE>
    </BOOK>
    <BOOK>
      <TITLE>Moby Dick</TITLE>
      <AUTHOR>Herman Melville</AUTHOR>
      <BINDING>copertina rigida</BINDING>
      <PAGES>724</PAGES>
      <PRICE>EURO 9,95</PRICE>
    </BOOK>
    <BOOK>
      <TITLE>Il gatto con gli stivali</TITLE>
      <AUTHOR>Charles Perrault</AUTHOR>
      <BINDING>tascabile economico</BINDING>
      <PAGES>120</PAGES>
      <PRICE>EURO 4,95</PRICE>
    </BOOK>
  </BIBLIOTECA>

/* Nome file: Biblioteca01.css */
BOOK
{display:block;
margin-top:14pt;
font-size:12pt}

TITLE
{font-weight:bold}
AUTHOR
{font-style:italic}
```

