

XML eXtensible Markup Language

Massimo Martinelli

Massimo.Martinelli@isti.cnr.it



Consiglio Nazionale delle Ricerche - CNR
Istituto di Scienza e Tecnologie della Informazione - ISTI



Cosa è XML

- "eXtensible Markup Language, abbreviato XML, descrive una classe di dati chiamati documenti XML e descrive parzialmente il comportamento dei programmi che li elaborano"
- è un linguaggio di marcatura (basato su markup) estensibile realizzato per poter utilizzare in modo semplice i documenti strutturati.
Sviluppato dal W3C, il *World Wide Web Consortium*
- *Markup (marca, etichetta)*: tutto ciò che ha un significato speciale che deve essere ben caratterizzato, reso esplicito anche identificatore, simbolo o altro espediente per distinguere un elemento da altri simili,
può indicare l'inizio o la fine di un oggetto
- Esempi di markup: *testo in italico*, testo sottolineato
- In XML tutto ciò che è compreso tra i caratteri "<" e ">" (*angled brackets*) è considerato *markup*, viene detto anche *tag* (etichetta)
Esempio:

<nome>

Estensibilità

- XML non ha *tag* predefiniti, è estensibile, consente di definire nuovi linguaggi, è un metalinguaggio. Ci permette di definire altri metalinguaggi, anche un nostro personale metalinguaggio
- Metalinguaggio: nella logica formale, linguaggio impiegato nello studio di un linguaggio oggetto. Può essere o non essere formalizzato e a sua volta può essere oggetto di indagine che si attua mediante un metametalinguaggio.
Detto anche linguaggio di descrizione del linguaggio

Metainformazione

A cosa può servirci una metainformazione ?

Esempio: cosa rappresenta la seguente informazione ?

pol 350,75 379,161 469,161 397,215
423,301 350,250 277,301 303,215
231,161 321,161

Visualizza !



Linguaggio

- Se vogliamo che anche gli altri capiscano il nostro metalinguaggio è necessario stabilire alcune regole:
 - dichiarare che cosa costituisce un markup
 - dichiarare esattamente che cosa significa il nostro markup
- Un markup language è quindi un insieme di regole
- *Anche HTML è un markup language*
- L'insieme delle regole di HTML è contenuto in un documento (separato dal file .html)
il DTD HTML (*Document Type Definition*)
incorporato nel browser, invisibile all'utente.

XML - eXtensible Markup Language

- Prima bozza di XML:
novembre 1996
- Specifica attuale:
<http://www.w3.org/TR/REC-xml>
- Alcune traduzioni in italiano:
<http://www.w3c.it>

I documenti XML

- I documenti XML sono costituiti da unità (entità),
- Entità: contengono dati analizzabili e non analizzabili
- Dati analizzabili: costituiti da caratteri, alcuni dei quali formano i character data, e alcuni i markup.
- Markup: usato per strutturare logicamente il documento e per l'organizzazione della memorizzazione.
- XML fornisce un meccanismo per imporre dei vincoli sull'organizzazione di memorizzazione e sulla struttura logica.
- Un modulo software detto **processore XML** viene usato per leggere documenti XML e fornire l'accesso al loro contenuto e alla loro struttura.
- Processore XML lavora per conto di un altro modulo: **l'applicazione**.
- La specifica XML descrive il comportamento che un processore XML deve tenere nel leggere i dati XML e l'informazione che deve fornire all'applicazione.

Perchè un nuovo linguaggio ?

Limiti di HTML

Cosa è questo ?

```
<td> 12 </td>
```

- Il numero civico di una via ?
- Il numero di telefono per ottenere informazioni sugli abbonati ?
- Entrambe le cose ?
- Nessuna delle due ?

Un semplice markup con HTML

```
<p> <b> Sig. Mario Rossi </b>  
<br>  
Via Verdi, 12  
<br>  
56100, Pisa
```

Visualizzazione di markup HTML

Sig. Mario Rossi
Via Verdi, 12
56100, Pisa

Interpretazione di HTML

Il nostro algoritmo per trovare il numero civico:

Se un paragrafo contiene due tag

allora la prima parola dopo la prima virgola dopo il primo tag
 è il numero civico.

Un semplice markup XML

```
<business-card>  
  <persona>  
    <titolo> Sig. </titolo>  
    <nome> Mario </nome>  
    <cognome> Rossi </cognome>  
  </persona>  
  <indirizzo>  
    <strada> Via Verdi </strada>  
    <numero-civico> 12 </numero-civico>  
    <cap> 56100 </cap>  
    <città> Pisa </città>  
</business-card>
```

Visualizzazione di markup XML

Sig. Mario Rossi
Via Verdi, 12
56100, Pisa

- XML può essere visualizzato nello stesso modo di HTML

Visualizzazione di markup XML

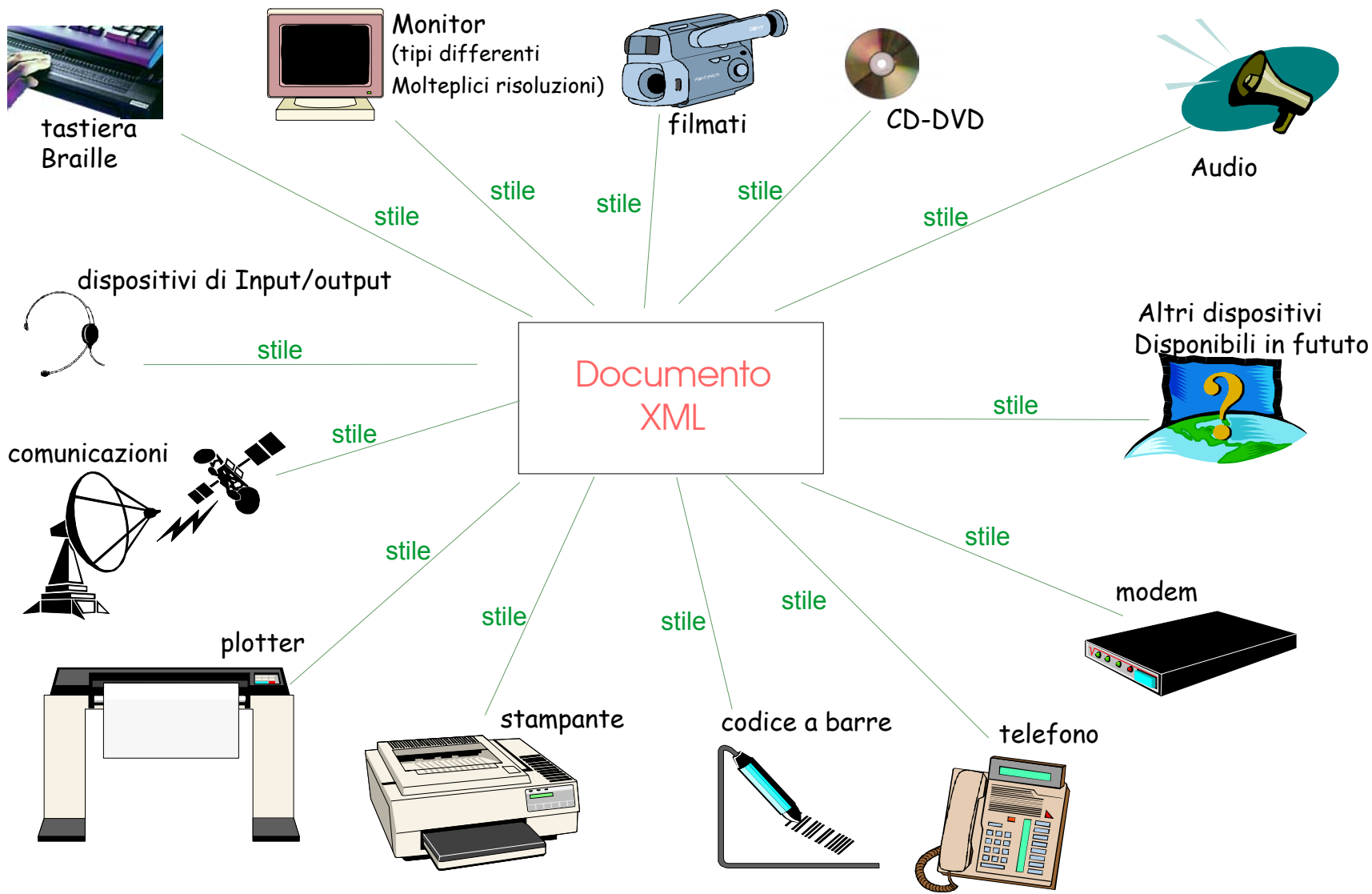
Il markup XML può essere visualizzato anche in questo modo:

Sig. Mario Rossi

Via Verdi, 12

56100, **Pisa**

Visualizzazione di markup XML



Interpretazione di XML

Un algoritmo migliore e più semplice per trovare il numero civico:

il numero civico è il contenuto del tag `<numero-civico>`

Contenuto contro rappresentazione

HTML ci dice come rappresentare un documento ipertestuale su Web
(difficilmente su un altro media, ad esempio su carta)

XML ci dice cosa contiene un documento

Non è sufficiente migliorare HTML

Ricapitolando: limiti di HTML

- non ci dice nulla sul contenuto del documento
- non permette di estendere il linguaggio con tag personali
- limitato come prodotto di pubblicazione
- limitato come ipertesto
- limitato come elaborazione
- non supporta dati strutturati -> inefficiente per i motori di ricerca

Serve un linguaggio semplice, flessibile

HTML non verrà comunque sostituito, almeno nel più immediato futuro, perché offre il metodo più semplice per pubblicare informazioni sul Web

Le componenti di XML

Problema attuale: scambio di documenti
Formati proprietari difficilmente scambiabili

XML studiato per facilitare scambi di dati anche tra applicazioni di tipo diverso (es.: i database e i word processor).

Documento facilmente interpretabile
tre parti fondamentali da tenere distinte:

- il contenuto;
- le specifiche relative agli elementi, la struttura (DTD);
- le specifiche relative alla rappresentazione, lo stile (*Stylesheet*).

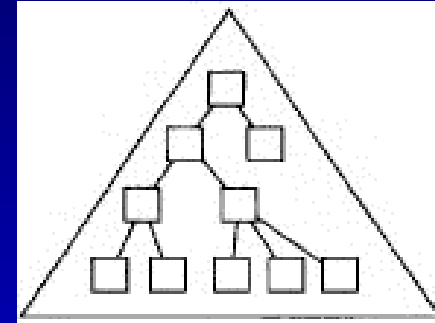
Le componenti di XML

Tre parti fondamentali distinte : 1) il contenuto; 2) le specifiche relative agli elementi, la struttura (DTD); 3) le specifiche relative alla rappresentazione, lo stile (*Stylesheet*).

Contenuto

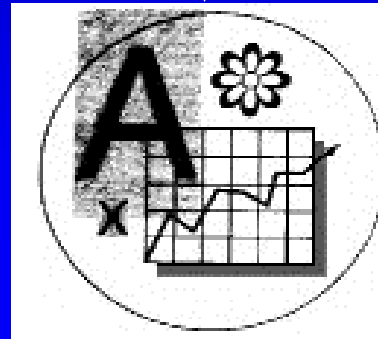
"La Discovery stava accelerando verso Giove lungo un'orbita complessa calcolata alcuni mesi prima dagli astronomi sulla Terra e controllata costantemente da Hal"
A.C. Clarke

Struttura



Formattazione

Specifiche di formattazione dell'output



Processo completo di codifica XML

Rappresentazione (Stile)

Il documento XML

- Ciascun documento XML ha una struttura logica e fisica.
- Fisicamente, il documento è composto di unità dette entità. Un entità può indirizzare altre entità per includerle nel documento. Un documento inizia con una "radice" o entità documento.
- Logicamente, il documento è composto di dichiarazioni, elementi, commenti, riferimenti a caratteri, e istruzioni di elaborazione, ciascuno dei quali è indicato nel documento da espliciti markup. Le struttura logica e quella fisica devono annidarsi propriamente

Come si presenta un documento XML

- Anagrafe.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<anagrafe>
```

```
  <cie numerocarta="XX123456YY">
```

```
    <cognome>Rossi</cognome>
```

```
    <nome>Mario</nome>
```

```
    <datanascita>
```

```
      <giorno>20</giorno>
```

```
      <mese>03</mese>
```

```
      <anno>2003</anno>
```

```
    </datanascita>
```

```
    <luogonascita>
```

```
      <comune>Livorno</comune>
```

```
      <provincia>LI</provincia>
```

```
    </luogonascita>
```

```
  </cie>
```

```
</anagrafe>
```

Un ipotetico colloquio tra un Ente (es. Ministero degli Interni, Prefettura, Camera di Commercio,...) e l'Anagrafe

Uno degli obiettivi di progettazione di XML è che sia in un formato leggibile dall'uomo, per intendersi non può essere in formato binario, e dovrebbe essere ragionevolmente chiaro.

Prologo

Ogni documento XML inizia con un prologo

- contiene una dichiarazione di versione
- l'insieme di caratteri utilizzato (character set)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

...



Documento conforme alla
versione 1.0 di XML



Documento che usa caratteri
appartenenti alla codifica ISO-8859-1
"Western European"

Internazionalizzazione

- XML supporta Unicode:
 - Rende possibile l'utilizzo contemporaneo di differenti alfabeti senza difficoltà
 - La codifica dei caratteri identifica gli stessi: non devo utilizzare codifiche particolari per far riconoscere i caratteri, uso direttamente i caratteri che verranno riconosciuti poiché dichiaro il set di caratteri
 - UTF-8: 8-bit Unicode
 - UTF-16: 16-bit Unicode

Elementi

- Radice (root):
elemento che racchiude tutti gli altri

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<anagrafe>
```

```
...
```

```
</anagrafe>
```

Regole per i tag

- Ogni *tag* di apertura deve avere un corrispondente *tag* di chiusura
`</nome_elemento>`
- Forma concisa per elementi senza contenuto:
`<nome_elemento />`
forme equivalenti
`<nome_elemento attr="valore"></nome_elemento>`
`<nome_elemento attr="valore" />`
- Esempi HTML
`
`. (`
</br>`)

``

Regole per i tag

I tag devono essere nidificati correttamente

~~Non permesso~~

~~<aa>
 <bb>
</aa>
 </bb>~~

Corretto!

<aa>
 <bb>
 </bb>
</aa>

Markup XML

- XML è case sensitive


le lettere maiuscole e quelle minuscole sono interpretate differentemente

<nome> ≠ <Nome> ≠ <NOME>

- i *tag* XML si scrivono in minuscolo
- Alcuni caratteri e sequenze di caratteri sono riservati, pertanto non si possono utilizzare nei nomi di *tag* (`%`, `xml`, ...).

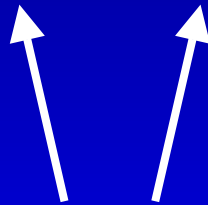
Attributi

<cie numerocarta="XX123456YY">



- Sintassi:

<elemento attributo="valore">contenuto</elemento>

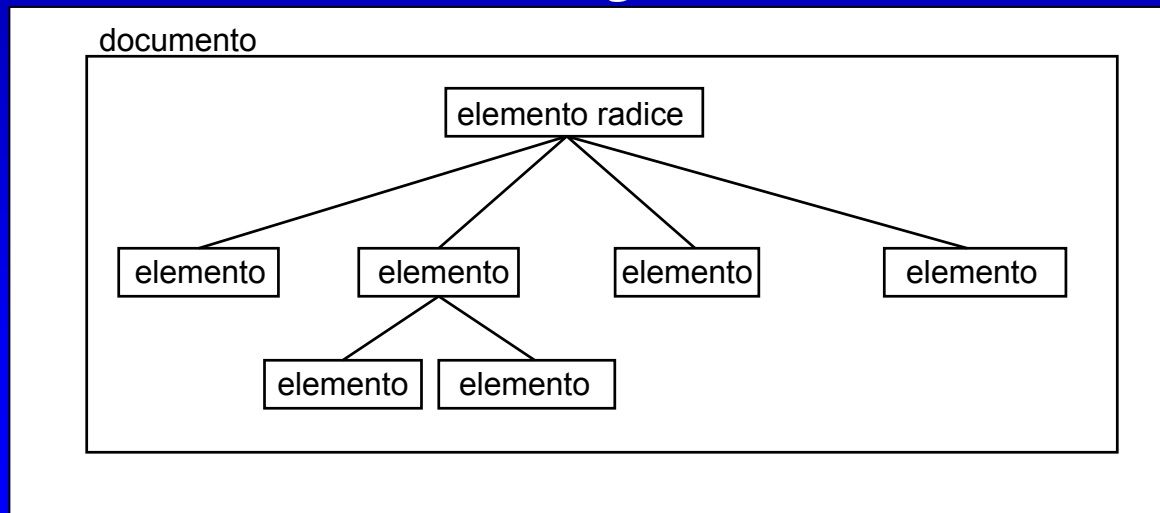


Quotare gli attributi

DTD - Document Type Definition

- Contiene le regole di definizione dei tag
- Indica gli elementi e il loro ordine all'interno del documento XML
- Il suo uso non è obbligatorio; ne è consigliato l'utilizzo
- Può essere interno o esterno al documento XML
- Il suo nome per convenzione corrisponde a quello dell'elemento radice

Struttura gerarchica



Un esempio di DTD

```
<!ELEMENT anagrafe (cie+)>
<!ELEMENT cie (cognome, nome, datanascita, luogonascita,
  luogoresidenza?, infoaggiuntiva*)>
<!ATTLIST cie
  numerocarta ID #REQUIRED>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT datanascita (giorno, mese, anno)>
<!ELEMENT giorno (#PCDATA)>
<!ELEMENT mese (#PCDATA)>
<!ELEMENT anno (#PCDATA)>
<!ELEMENT luogonascita (comune, provincia, statoestero?)>
<!ELEMENT comune (#PCDATA)>
<!ELEMENT provincia (#PCDATA)>
<!ELEMENT statoestero (#PCDATA)> <!-- solo se != Italia -->
<!ELEMENT luogoresidenza (comune, provincia, statoestero?)>
  <!-- solo se diversa da quella di nascita -->
<!ELEMENT infoaggiuntiva (#PCDATA)>
```


Riferimento alla DTD

- Internamente al file XML: dichiarazione DOCTYPE:

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

```
<!DOCTYPE elemento_radice [
```

```
  definizione elementi, attributi, entità, notazioni
```

```
 ]>
```

```
< anagrafe >
```

```
...
```

```
</ anagrafe >
```

- in caso di DTD esterna:
dichiarazione DOCTYPE (document type declaration) nel
file XML

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

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

Dichiarazione di elementi

- Successivamente alla dichiarazione DOCTYPE si dichiarano gli elementi
- SINTASSI:
**<!ELEMENT nome_elemento
(nomi_degli_elementi_permessi)>**

Ordine degli elementi

- Osserviamo la dichiarazione dell'elemento "cie" nel nostro esempio:
<!ELEMENT cie (cognome, nome, datanascita, luogonascita, luogoresidenza?, infoaggiuntiva*)>
(ELEMENT CONTENT)
- si può notare che è composto dagli elementi "cognome", "nome", "datanascita", "luogonascita", "luogoresidenza", "infoaggiuntiva"
- Questi elementi sono separati da una virgola pertanto devono essere presenti nel file anagrafe.xml in questo stesso ordine.
- l'elemento *infoaggiuntiva* è seguito dal carattere "*", questo significa che ce ne potrebbero essere 0, 1 o più di uno.
- Inoltre l'elemento *luogoresidenza* è seguito dal carattere "?" ciò significa che potrà essere presente zero o una volta.
- Prendiamo in esame le seguenti dichiarazioni (nuovo esempio):
<!ELEMENT persona (nome|cognome)>
<!ELEMENT persona (nome|cognome|email)+>
- Nel primo caso il carattere "|" sta a significare che l'elemento "*persona*" sarà costituito dall'elemento "*nome*" oppure dall'elemento "*cognome*";
- nel secondo caso in cui è presente un carattere "+", da almeno 1 ma anche da più elementi "*nome*", "*cognome*", "*email*" in qualsiasi ordine.

Occorrenza degli elementi

Elemento	(1)
Elemento ?	(0,1)
Elemento *	(0,1,+)
Elemento +	(1,+)
Elemento1 , Elemento2	(and, stesso ordine)
Elemento1 Elemento2	(or)

Tipi

<!ELEMENT cognome (#PCDATA)>

- l'elemento cognome potrà essere composto da qualsiasi testo o altro carattere che non sia un markup o " , & oppure]]

PCDATA = *Parsed Character Data*

CDATA = *Character Data*

Sezioni CDATA

- Sezione CDATA: un modo conveniente per rendere più semplice la vita agli autori di documenti XML.
- Supponiamo di voler includere un esempio XML in un documento XML, ci sono due modi per farlo:
 - `<tag>ciao !</tag>`
 - `<![CDATA[<tag>ciao !</tag>]]>`
- Con CDATA è possibile inserire frammenti di codice XML all'interno di altri documenti XML senza dover preoccuparsi di usare le entità per i caratteri "<" e "&" (escaping)

Documento ben-formato, valido

- Un documento XML si dice "ben formato" quando:
 - contiene almeno un elemento;
 - esiste un *tag* unico di apertura e di chiusura che contiene l'intero documento; (radice, root)
 - tutti i tag sono nidificati
 - tutte le entità sono dichiarate.

- Un documento si dice "valido" quando
 - contiene una DTD e rispetta le regole definite in essa.

ANY

- Oltre a #PCDATA e element content (elementi che contengono altri elementi) esistono altri modelli di contenuto per gli elementi:
- ANY:
specifica che il contenuto di un elemento può essere una qualsiasi sequenza di elementi (definiti nella DTD) in un qualunque ordine e numero di occorrenza.

<!ELEMENT contenitore ANY>

EMPTY

Specifica che il contenuto di un elemento è vuoto

```
<!ELEMENT br EMPTY>
```

Contenuto Misto (Mixed)

- Specifica che il contenuto di un elemento può essere una qualsiasi sequenza di caratteri o di elementi figlio (definiti nella DTD) o una sequenza mista di elementi e caratteri, in qualunque ordine e numero di occorrenza.
- I tipi degli elementi figlio possono essere vincolati, ma non l'ordine o il numero delle loro occorrenze

```
<!ELEMENT p (#PCDATA|elemento1|elemento2)*>
```

Dichiarazione unica di tipo di elemento

- Nessun tipo di elemento può essere dichiarato più di una volta.

Attributi

- Gli attributi sono usati per associare coppie di nome-valore agli elementi. Le specifiche di attributo possono apparire solo all'interno dei tag-di-inizio e dei tag degli elementi vuoti
- Dichiarazioni di lista di attributi (Attribute-List)

**<!ATTLIST nome_elemento
 nome_attributo tipo_di_attributo
valore_di_default>**

- I tipi di attributo possono essere
 - CDATA (qualsiasi testo)
 - Un insieme di tipi "tokenized" (ID, IDREF)
 - una lista di valori (tipo enumerato)

Tipi di attributi tokenized

- **ID** : i valori devono essere univoci
- Nel nostro esempio ogni elemento *cie* avrà un codice univoco (ID).
 <!ATTLIST *cie* numerocarta ID #REQUIRED>
- Nessun "element type" può avere specificato più di un attributo di tipo ID.
- Il valore di un attributo è obbligatorio quando è specificata l'opzione #REQUIRED (nel nostro esempio il codice dell'elemento *cie* è obbligatorio), può non avere un codice specificato se l'opzione è #IMPLIED, oppure il solo valore specificato è valido con l'opzione #FIXED.
- Un attributo ID deve avere un default che sia dichiarato #IMPLIED o #REQUIRED.

Vincoli di validità degli attributi

- **IDREF**
i valori IDREF devono essere uguali al valore di qualche attributo ID dichiarata nel DTD.
- **Nome di entità**
deve corrispondere al nome di una entità unparsed dichiarata nel DTD.
- **Attributi enumerati**
possono assumere uno dei valori tratti da una lista fornita nella dichiarazione.

Riepilogo Attributi

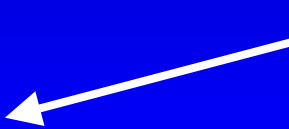
```
<!ATTLIST nome_elemento
  nome_attributo1 tipo_attributo valore_di_default #REQUIRED >
  (obbligatorio)
```

```
<!ATTLIST nome_elemento
  nome_attributo2 tipo_attributo #IMPLIED >
  (opzionale)
```

```
<!ATTLIST nome_elemento
  nome_attributo3 tipo_attributo valore_di_default #FIXED >
  (obbligatorio con valore prefissato)
```

- Esempi di dichiarazioni di liste di attributi

```
<!ATTLIST termdef
  id ID #REQUIRED
  name CDATA #IMPLIED>
<!ATTLIST list
  type (verde|bianco|rosso) "rosso">
<!ATTLIST form
  method CDATA #FIXED "POST">
```



default

Entità

- Si possono definire entità nel caso ci siano elementi ripetuti spesso: ad esempio
 - `<!ENTITY XML "eXtensible Markup Language">`
- ogni occorrenza di `&XML;` (nel file XML) sarà sostituita in fase di visualizzazione dalla stringa
 - `" eXtensible Markup Language "`
- Le entità si possono utilizzare per rappresentare caratteri riservati come ad esempio `"<"` o `">"` (`<`, `>`), o possono essere usate per far riferimento a documenti esterni nel caso il documento XML non sia composto da un unico file:
 - `<!ENTITY introduzione SYSTEM "introduzione.txt">`
- Possono esserci anche entità parametriche
 - `<!ENTITY % [nome] "[nomi]">`
- ad esempio:
 - `<!ENTITY % headings "H1 | H2 | H3 | H4">`
 - `<!ENTITY BODY (%headings | P | DIV | BR)*>`

Parser

- Analizzano i documenti XML, due tipi:
 - Validanti consentono di verificare la congruità del documento, la validità rispetto alle regole imposte dalla DTD
 - Non Validanti permettono di verificare solo la correttezza formale dei documenti (well-formedness)
- Utili entrambi, utilizzo in contesti differenti (a volte non serve verificare la validità ma solo la correttezza)

Dichiarazione di Notazione

- Le dichiarazioni di notazione servono per identificare specifici tipi di dati binari esterni, come ad esempio i file in formato "gif"

```
<!NOTATION gif87a SYSTEM "GIF">
```

Spazio (Whitespace)

- **Trattamento degli spazi** (spazi, tabulazioni, e interruzioni di linea)
- Un attributo speciale chiamato **xml:space** può essere associato ad un elemento per segnalare un'intenzione che in quell'elemento, gli spazi bianchi dovrebbero essere preservati dalle applicazioni.
- I possibili valori sono solo "default" e "preserve".
- Esempio:
<!ATTLIST poem xml:space (default|preserve) 'preserve' >
- Il valore "preserve" indica l'intento che le applicazioni conservino tutti gli spazi bianchi. Questa intenzione dichiarata si estende a tutti gli elementi interni al contenuto dell'elemento in cui è specificata, a meno che non sia soprascritta da un'altra istanza dell'attributo xml:space.
- Il valore "default" segnala che i modi di default di elaborazione degli spazi bianchi da parte dell'applicazione sono accettabili per questo elemento;

Istruzioni di Elaborazione (Processing Instructions)

<? istruzione ?>

- Istruzioni che possono essere elaborate da una applicazione
(in un modo riconosciuto dall'applicazione)
- Anche questa è una istruzione di elaborazione:
<?xml version="1.0" encoding="ISO-8859-1" ?>

Commenti

- `<!-- testo del commento -->`

Il testo del commento non può contenere i caratteri `--`

(altrimenti il parser non "capisce" dove termina)

Identificazione del linguaggio

- Nell'elaborazione di un documento, spesso è utile identificare il linguaggio naturale o formale in cui il contenuto è scritto. Uno attributo speciale chiamato **xml:lang** può essere inserito nei documenti per specificare il linguaggio usato nei contenuti e nei valori degli attributi di ogni elemento di un documento XML.
- Nei documenti validi, questo attributo, come ogni altro, se è usato deve essere dichiarato. I valori di questo attributo sono gli identificatori di linguaggio come definiti da [IETF RFC 1766], "Tag per la identificazione dei linguaggi":

```
<p xml:lang="en">The quick brown fox jumps over the lazy dog.</p>
```

```
<p xml:lang="en-GB">What colour is it?</p>
```

```
<p xml:lang="en-US">What color is it?</p>
```

```
<!ATTLIST note xml:lang NMTOKEN 'en'>
```

Impiego di XML

Scambio dati
Publishing

- Principali tecnologie associate ad XML

CSS	Privacy and P3P	XLink
DOM	RDF	XML Protocol
HTML	Semantic Web	XQuery
HTTP	SMIL	XML Schema
UNICODE	SVG	XML Signature
MathML	VoiceXML	XPath
Micropayments	WAI	XSL & XSLT
Mobile	XForms	...
PICS	XHTML	
	XML Encryption	

Moltissime applicazioni oggi usano XML per gli usi più svariati

- metadati x RPM (pacchetti linux)
- file di configurazione (tomcat,...)
- file di proprietà (winamp,...)
- ChessML (scacchi ;-P)
- ...

Perché è importante XML

- Problemi principali da risolvere sul Web: lentezza, difficoltà trovare informazioni.
- XML permette di strutturare i documenti e di associare una sintassi.
- Riduzione del traffico di rete, maggiore sviluppo applicazioni client.
- XML supporta UNICODE, un grande numero di set di caratteri può essere utilizzato.
- XML può essere usato come formato di scambio per documenti elettronici ed applicazioni.
- Indipendente da hardware e software
- Struttura contenente di manipolare i dati in modo più semplice ed efficiente.
- Quando registriamo informazioni vogliamo essere sicuri di poterle riutilizzare in futuro.
(es. word 1, word 2, word per mac, word per pc, incompatibilità, perdita di informazioni)
- Un documento XML è in formato leggibile dall'uomo.
- Documenti strutturati e metainformazioni consentono motori di ricerca più accurati
("ALBERO BINARIO")
- Sistemi standard di metadati: Resource Description Framework.
- XML è estensibile, non ha tag predefiniti.
- Consente di creare linguaggi standard ed estendibili, per campi di applicazione
(medicina, elettronica, matematica, musica, ...)
- Con il DOM + semplice scrivere un programma che trova gli elementi e li usa
- I collegamenti offrono nuove possibilità. (punti precisi, più destinazioni, database centralizzato -> maggiore maneggevolezza, più semplice controllare corrispondenza collegamenti).
- Offre una ottima capacità di rappresentare dati complessi (notazioni matematiche, interfacce grafiche)
- Visualizzare documento su media differenti in modi diversi senza doverlo riscrivere ogni volta.
- XSL offre meccanismi per rappresentare e manipolare il documento, buona capacità di rappresentare dati complessi (notazioni matematiche, interfacce grafiche); sequenze, cicli e condizioni
(N-regime)

Riferimenti

- 1 <http://www.w3.org/XML> la home page di XML sul sito del W3C
- 2 <http://www.w3.org> La home page del W3C
- 3 <http://www.w3.org/TR/> Standard, Draft, Note W3C
- 4 "The Annotated XML Specification" Tim Bray
<http://www.xml.com/axml/testaxml.htm>
- 5 <http://www.xml.org>
- 6 <http://www.xml.com>
- 7 <http://www.microsoft.com/xml>
- 8 <http://www.mozilla.org>
- 9 <http://www.netscape.com>
- 10 <http://www.xmlsoftware.com>

<http://www.iei.pi.cnr.it/~Martinelli/>

http://www.iei.pi.cnr.it/Personal/Martinelli/XML/Doc/XML-A_Technical_Introduction.PDF

Il Gruppo XML Italia

Le mailing-list del gruppo XML-Italia:

org@xml.it riservata all'organizzazione

xml@xml.it aperta, dedicata alle attività tecnico-informative

consultabili anche su web:

<http://listserv.xml.it/org.html>

<http://listserv.xml.it/xml.html>

<http://www.xml.it> Il sito di XML Italia





- <http://www.w3c.it>
- Alcune specifiche tradotte in italiano