La sintassi di XML

Fabio Vitali

Sommario

- Sezione I: visione generale
  - Elementi di un documento XML
  - Formato di un documento XML
- Sezione II: Il contenuto di un DTD
  - Definizione di elementi
  - Definizione di attributi
  - Definizione di entità generali
  - Definizione di entità parametriche
  - Altri aspetti di XML

XML 1.0

- Una raccomandazione W3C del 10 febbraio 1998.
- È definita come un sottoinsieme di SGML
- URL ufficiale: `http://www.w3.org/TR/REC-xml`
- Traduzione ufficiale in italiano:
  `http://www.iat.cnr.it/xml/REC-xml-19980210-it.html`
- Molto più formalizzata della grammatica di SGML, usa una notazione formale, ***Extended Backus-Naur Form***.

XML e Unicode

XML (come Java) abbandona completamente ASCII e le codifiche ad un byte, e si basa direttamente su Unicode.

Questo porta a due vantaggi nei riguardi dell'internazionalizzazione:

- È possibile scrivere documenti misti, senza ricorrere a trucchi strani per identificare la parte che usa un alfabeto dalla pare che ne adopera un altro.
- Un documento scritto in un linguaggio non latino non deve basarsi su parametri esterni per essere riconosciuto come tale, ma la codifica stessa dei caratteri lo identifica.

I componenti di XML

Un documento XML contiene una varietà dei seguenti componenti

- Elementi
- Attributi
- Entità
- Testo (detto anche #PCDATA)
- Commenti
- Processing Instructions

Elementi, Attributi, Entità, #PCDATA, Commenti, Processing Instructions

- Gli elementi sono le parti di documento dotate di un senso proprio.
- Il titolo, l'autore, i paragrafi del documento sono tutti elementi.
- Un elemento è individuato da un tag iniziale, un contenuto ed un tag finale.
- **Non confondere i tag con gli elementi!**

```
<TITOLO>Tre uomini in barca</TITOLO>
```


Elementi, **Attributi**, Entità, #PCDATA, Commenti, Processing Instructions

- Gli attributi sono informazioni aggiuntive sull'elemento che non fanno effettivamente parte del contenuto (meta-informazioni).
- Essi sono posti dentro al tag iniziale dell'elemento. Tipicamente hanno la forma nome=valore

```
<romanzo file="threemen.sgm">…</romanzo>
<capitolo N="1">Capitolo primo</capitolo>
```


Elementi, Attributi, **Entità**, #PCDATA, Commenti, Processing Instructions

- Le entità sono frammenti di documento memorizzati separatamente e richiamabili all'interno del documento.
- Esse permettono di riutilizzare lo stesso frammento in molte posizioni garantendo sempre l'esatta corrispondenza dei dati, e permettendo una loro modifica semplificata.

```
Oggi &egrave; una bella giornata.
Come dice &FV;: “divertitevi!”
```


Elementi, Attributi, Entità, #PCDATA, Commenti, Processing Instructions

- Rappresenta il contenuto vero e proprio del documento.
- Esso corrisponde alle parole, gli spazi e la punteggiatura che costituiscono il testo.
- Viene anche detto #PCDATA (Parsed Character DATA) perché XML usa questo nome per indicare il contenuto di elementi di testo.

Elementi, Attributi, Entità, #PCDATA, **Commenti**, Processing Instructions

I documenti XML possono contenere commenti, ovvero note da un autore all'altro, da un editore all'altro, ecc.

Queste note non fanno parte del contenuto del documento, e le applicazioni XML li ignorano.

Sono molto comodi per passare informazioni tra un autore e l'altro, o per trattenere informazioni per se stessi, nel caso le dimenticassimo.

```
<!-- Questa parte è ignorata da XML -->
```


Elementi, Attributi, Entità, #PCDATA, Commenti, **Processing Instructions**

- Le **processing instructions** (PI) sono elementi particolari (spesso di senso esplicitamente procedurale) posti dall'autore o dall'applicazione per dare ulteriori indicazioni su come gestire il documento XML nel caso specifico
- Per esempio, in generale è l'applicazione a decidere quando cambiare pagina. Ma in alcuni casi può essere importante specificare un comando di cambio pagina (oppure tutti i cambi pagina di un documento già impaginato).

```
<?NEWPAGE?> .
```


I documenti XML

Un documento in un linguaggio di markup definito sulla base di XML è composto dalle seguenti tre parti:

- Dichiarazione XML (opzionale)
  - La specifica delle opzioni attivabili nel documento XML
- Document Type Declaration (opzionale)
  - Le regole di validità a cui il documento deve sottostare
- Istanza del documento
  - Il contenuto ed il markup effettivo del documento

Dichiarazione XML
Document Type Declaration
Document Instance

```
<?XML version="1.0" encoding="UTF-16" standalone="yes" ?>
```

La dichiarazione XML specifica le caratteristiche opzionali del documento in questione. Poiché esse sono ridotte al minimo, la dichiarazione XML è brevissima.

Esistono esattamente tre valori che possono essere messi in una dichiarazione XML:

- Il parametro "version" identifica quale versione di XML si sta usando. Ora l'unico valore possibile è "1.0". Necessario.
- Il parametro "encoding" permette di specificare, se il dubbio può sorgere, quale codifica di caratteri viene usata per il documento. Facoltativo.
- Il parametro "standalone" permette di specificare se le informazioni necessarie per valutare e validare il documento sono interne o se ne esistono anche di esterne. Facoltativo.

Dichiarazione XML
Document Type Declaration
Document Instance

- La dichiarazione del tipo del documento serve a specificare le regole che permettono di verificare la correttezza strutturale di un documento.
- Vengono cioè elencati [i file che contengono] gli elementi ammissibili, il contesto in cui possono apparire, ed altri eventuali vincoli strutturali.
- Nella terminologia XML, si parla di modellare una classe (cioè una collezione omogenea) di documenti attribuendogli un tipo.

Dichiarazione XML
Document Type Declaration
Document Instance

L’istanza del documento è quella parte del documento che contiene il testo vero e proprio, dotato del markup appropriato.

Le applicazioni XML sono in grado di verificare se l’istanza del documento segue le regole specificate nel DTD, e a identificare le violazioni.

Documenti ben formati o validi

XML distingue due tipi di documenti rilevanti per le applicazioni XML: i documenti ***ben formati*** ed i documenti ***validi***.

In SGML, un DTD è necessario per la validazione del documento. Anche in XML, un documento è **valido** se presenta un DTD ed è possibile validarlo usando il DTD.

Tuttavia XML permette anche documenti **ben formati**, ovvero documenti che, pur essendo privi di DTD, presentano una struttura sufficientemente regolare e comprensibile da poter essere controllata.

Documenti XML ben formati

Un documento XML si dice ben formato se:

- Tutti i tag di apertura e chiusura corrispondono e sono ben annidati
- Esiste un elemento radice che contiene tutti gli altri
- I tag vuoti (senza contenuto) utilizzano un simbolo speciale di fine tag: `<vuoto/>`
- Tutti gli attributi sono sempre racchiusi tra virgolette
- Tutte le entità sono definite.

Parser validanti e non validanti

- Il cuore di un applicazione XML è il parser, ovvero quel modulo che legge il documento XML e ne crea una rappresentazione interna utile per successive elaborazioni (come la visualizzazione).
- Un parser validante, in presenza di un DTD, è in grado di verificare la validità del documento, o di segnalare gli errori di markup presenti.
- Un parser non validante invece, anche in presenza di un DTD è solo in grado di verificare la buona forma del documento.
- Un parser non validante è molto più semplice e veloce da scrivere, ma è in grado di fare meno controlli. In alcune applicazioni, però, non è necessario validare i documenti, solo verificare la loro buona forma.

Esempi di sviluppo del DTD

Ci sono quattro caratteristiche dei DTD XML che vale la pena considerare a fondo:

- La dichiarazione di tipo (**<!DOCTYPE … >)**
- Elementi **<A> … </A>**
- Attributi <A **HREF="http://…"**> … </A>
- Entità: ***à***

La dichiarazione di tipo

- Il **<!DOCTYPE ... >** è la dichiarazione del tipo di documento. Essa permette alle applicazioni XML di determinare le regole sintattiche da applicare alla verifica e validazione del documento.
- La dichiarazione non è, ma contiene o fa riferimento alla Document Type Definition, o DTD, ove vengono elencati gli elementi validi e i loro vincoli.
- Il DTD può essere posto in un file esterno, internamente al documento, o in parte esternamente ed in parte internamente.

Dichiarazione del DTD: <!DOCTYPE ... >

```
1 <!DOCTYPE mydoc SYSTEM “document.dtd“>
2 <!DOCTYPE mydoc [
            <!ELEMENT …
  ]>
3 <!DOCTYPE mydoc SYSTEM “document.dtd” [
            <!ELEMENT …
  ]>
```

- La prima forma di dichiarazione indica che il DTD è contenuto in un file esterno (per esempio, condivisa con altri documenti). Il DTD viene chiamato ***external subset*** perché è posto in un file esterno.
- La seconda forma precisa il DTD internamente (cioè nello stesso file), che quindi non può essere condiviso da altri file. Il DTD si chiama in questo caso ***internal subset***.
- La terza forma precisa una parte del DTD come contenuta in un file esterno (e quindi condivisibile con altri documenti), e una parte come propria del documento, e non condivisibile.

Specifica di elementi

```
<!ELEMENT nome content-model >
<!ELEMENT para (#PCDATA | bold)* >
```

Content-model: la specificazione formale del contenuto permesso nell'elemento, secondo una sintassi specifica di gruppi di modelli.

Specifica di elementi
Content model

Tramite il content model posso specificare quali sono gli elementi leciti all'interno di un elemento, in quale numero e quale posizione rispetto agli altri.

Ci sono cinque famiglie di content model:

- **Qualunque**: un elemento può contenere qualunque elemento definito nel DTD
- **Vuoto**: un elemento non ha contenuto, e spesso ha solo il tag iniziale.
- **Testo**: un elemento può contenere solo testo (#PCDATA)
- **Strutturato**: un elemento può contenere solo altri elementi
- **Misto**: un elemento può contenere sia testo che altri elementi. XML ha solo UNA forma ammessa di content model misto.

Specifica di elementi
Content model: ANY ed EMPTY

<!ELEMENT X ANY>

- L'elemento X può contenere qualunque altro elemento specificato nel DTD, o anche testo

<!ELEMENT X EMPTY>

- L'elemento X non può contenere niente. Debbo scrivere il tag con la sintassi tipica degli elementi vuoti: <X/>

Specifica di elementi
Content model testo e strutturato

```
<!ELEMENT X #PCDATA>
```

- L'elemento X può solo contenere testo. E' proibito mettere altri elementi al suo interno

```
<!ELEMENT X (Y, (W | Z)+ , K*)>
```

- L'elemento X può contenere solo elementi secondo la specifica data, usando i separatori e gli operatori di ripetizione specificati.

Specifica di elementi
Separatori

Separano specifiche determinando l'ordine o l'obbligatorietà:

- **',' (virgola)**: richiede la presenza di entrambe le specifiche nell'ordine precisato.
  Es.: **(a , b)**: ci devono essere sia a che b, e prima ci deve essere a e poi b.
- **'|' (barra verticale)**: ammette la presenza di una sola delle due specifiche.
  Es.: **(a | b)**: ci può essere o a, oppure b, ma solo uno di essi.

Specifica di elementi
Operatori di ripetizione (1)

Permettono di specificare se un gruppo può comparire esattamente una volta, almeno una volta, oppure zero o più volte.

- ***Niente***: la specifica precedente deve comparire esattamente una volta.
  Es.: c, (a, b): a e b devono comparire in quest'ordine esattamente una volta. È lecito solo: cab. Si dice che è una specifica ***richiesta*** e ***non ripetibile***.
- ***? (punto interrogativo)***: la specifica precedente può e può non comparire, ma solo una volta.
  Es.: c, (a , b)?: a e b possono comparire una volta, ma possono non comparire. Sono lecite: c, cab. Si dice che è una specifica ***facoltativa*** e ***non ripetibile***.

Specifica di elementi
Operatori di ripetizione (2)

- **+ *(più)***: la specifica precedente deve comparire almeno una volta. Es.: c, (a , b)+: a e b devono comparire almeno una volta, ma possono comparire anche più di una. Sono lecite: cab, cabab, cababababab, ma non c, ca, cb, cba, cababa. Si dice che è una specifica ***richiesta*** e ***ripetibile***.
- **** (asterisco)***: la specifica precedente deve comparire zero o più volte. Es.: c, (a, b)*: a e b possono comparire o no, a scelta e in numero libero. Sono lecite: c, cab, cabab, cababababab, ma non ca, cb, cba, cababa. Si dice che è una specifica ***facoltativa*** e ***ripetibile***.

Specifica di elementi
Content model misto

```
<!ELEMENT X (#PCDATA | Y | W | Z)*>
```

- L'elemento X può contenere sia testo sia altri elementi in maniera ed ordine specificati.
- L'unica forma accettabile di content model misto in XML è composta da una ripetizione di elementi alternativi, in cui il primo è #PCDATA. Cioè solo content model della forma qui sopra.

Un esempio semplice

Vogliamo descrivere il contenuto del mio portfolio di azioni. Invento un frammento di DTD per descrivere le mie azioni:

```
<!DOCTYPE stock SYSTEM “stock.dtd”>
<stock>
    <id>
        <nome>zacx corp</nome>
        <simbolo>ZCXM</simbolo>
    </id>
    <prezzo>28.875</prezzo>
    <quantita>1150</quantita>
</stock>
```


Il file “stock.dtd”

```
<!ELEMENT stock(id, prezzo, quantita)>
<!ELEMENT id (nome, simbolo) >
<!ELEMENT nome #PCDATA>
<!ELEMENT simbolo #PCDATA>
<!ELEMENT prezzo #PCDATA>
<!ELEMENT quantita #PCDATA>
```


Complichiamo l'esempio: ripetizioni ed alternative

Il portfolio è composto da più azioni. Di ogni azione posso possederne o meno. Se ne posseggo, allora registro il prezzo di acquisto, altrimenti il prezzo attuale.

```
<!DOCTYPE portfolio SYSTEM "portfolio.dtd">
<portfolio>
    <stock>
        <id><nome>zacx corp</nome>
            <simbolo>ZCXM</simbolo></id>
        <acquisto>28.875</acquisto>
        <quantita>1150</quantita>
    </stock>
    <stock>
        <id><nome>zaffymat inc</nome>
            <simbolo>ZFFX</simbolo></id>
        <prezzo>92.250</prezzo>
    </stock>
    <stock>
        <id><name>zysmergy inc</name>
            <simbolo>ZYSZ</simbolo></id>
        <prezzo>20.313</prezzo>
    </stock>
</portfolio>
```


Il file “portfolio.dtd”

```
<!ELEMENT portfolio (stock)+>

<!ELEMENT stock
      (id, (prezzo | (acquisto, quantita)))>

<!ELEMENT id (nome, simbolo) >

<!ELEMENT nome #PCDATA>

<!ELEMENT simbolo #PCDATA>

<!ELEMENT prezzo #PCDATA>

<!ELEMENT acquisto #PCDATA>

<!ELEMENT quantita #PCDATA>
```


Un altro esempio: content model misto

Un libro, fatto di capitoli e a loro volta di paragrafi.

```
<!DOCTYPE ROMANZO SYSTEM “romanzo.dtd”>
<ROMANZO>
  <TITOLO>Tre Uomini in Barca</TITOLO>
  <AUTORE>Jerome K. Jerome</AUTORE> <ANNO>1889</ANNO>
  <CAPITOLO>
    <TITOLO>Capitolo primo</TITOLO>
    <INDICE>
      <EL>Tre invalidi</EL><EL>Le sofferenze di George e Harris
      </EL><EL>La vittima di centosette malattie inguaribili</EL>
    </INDICE>
    <PARA>Eravamo in quattro: George, William Samuel Harris,
    io, e Montmorency. Standocene seduti in camera mia, fumavamo e
    parlavamo di quanto fossimo malridotti… <INCISO>malridotti,
    dal punto di vista della salute, <INCISO>intendo,</INCISO>
    naturalmente</INCISO>. </PARA>
    <PARA>Ci sentivamo tutti piuttosto giù di corda,</PARA>
  </CAPITOLO>…
</ROMANZO>
```


Il file “romanzo.dtd”

```
<!ELEMENT ROMANZO   (TITOLO, AUTORE, ANNO, CAPITOLO+)>
<!ELEMENT CAPITOLO  (TITOLO, INDICE, PARA+)>
<!ELEMENT TITOLO    (#PCDATA) >
<!ELEMENT AUTORE    (#PCDATA) >
<!ELEMENT ANNO      (#PCDATA) >
<!ELEMENT INDICE    (EL+) >
<!ELEMENT EL        (#PCDATA) >
<!ELEMENT PARA      (#PCDATA | INCISO) * >
<!ELEMENT INCISO    (#PCDATA | INCISO)* >
```


Specifica di attributi

La dichiarazione **<!ATTLIST… >** permette di definire una lista di attributi lecita ad un elemento XML dichiarato in precedenza.

```
<!ATTLIST nome
        nome-attributo-1  tipo-1     default-1
        nome-attributo-2  tipo-2     default-2
        nome-attributo-3  tipo-3     default-3
        …
>
```

Gli attributi sono informazioni aggiuntive poste insieme all'elemento. Tecnicamente non fa parte del contenuto del documento, ma descrive e specifica l'elemento.

Ci sono quattro famiglie di attributi importanti:

- Qualunque stringa è lecita
- E' possibile scegliere solo uno dei valori proposti (lista)
- Il valore deve essere unico su tutto il documento (ID)
- Il valore deve essere uguale a quello di un ID esistente

Specifica di attributi
Attributi stringa e lista

```
<!ATTLIST X
     att       CDATA     “uno”
>
```

- Il tag iniziale di X può contenere un attributo chiamato “att”. Qualunque stringa è lecita. Se non viene specificata una stringa, il valore definito per default è “uno”

```
<!ATTLIST X
     att  (uno|due|tre)  “due”
>
```

- Il tag iniziale di X può contenere un attributo chiamato “att”. Sono leciti solo i valori descritti. Se non viene specificata una stringa, il valore definito per default è “due”

Specifica di attributi
Attributi ID e IDREF

```
<!ATTLIST X
      att          ID              #IMPLIED
>
```

- Il tag iniziale di X può contenere un attributo chiamato “att”. Sono leciti solo valori unici su tutto il documento. L’elemento X assume identificabilità assoluta all’interno del documento: è un “luogo notevole”. Poiché il valore deve essere sempre diverso, non è possibile specificare un valore di default.

```
<!ATTLIST X
        att     IDREF           #IMPLIED
>
```

- Il tag iniziale di X può contenere un attributo chiamato “att”. I valori di “att” debbono essere uguali ad un valore di un attributo di tipo ID esistente da qualche parte nel documento.

Specifica di attributi
Valori di default negli attributi

I valori di default importanti in XML sono di quattro tipi:

- **Valore esplicito**: un stringa tra virgolette
- **Valore necessario**: la keyword #REQUIRED.
- **Valore facoltativo**: la keyword #IMPLIED.
- **Valore esplicito e non modificabile**: la keyword #FIXED e una stringa tra virgolette. Usata per scopi particolari. La vedremo per XLink.

Il primo esempio, più: attributi testo e lista

Ogni azione viene attribuita alla borsa a cui appartiene.

Ogni prezzo ha specificata la valuta.

```
<!DOCTYPE portfolio SYSTEM “portfolio.dtd”>
<portfolio>
    <stock exchange=“nyse”>
        <id><nome>zacx corp</nome>
            <simbolo>ZCXM</simbolo></id>
        <acquisto unit=“USD”>28.875</acquisto>
        <quantita>1150</quantita>
    </stock>
    <stock exchange=“nasdaq”>
        <id><nome>zaffymat inc</nome>
            <simbolo>ZFFX</simbolo></id>
        <prezzo unit=“USD”>92.250</prezzo>
    </stock>
    <stock exchange=“Frankfurt”>
        <id><name>zysmergy inc</name>
            <simbolo>ZYSZ</simbolo></id>
        <prezzo>20.313</prezzo>
    </stock>
</portfolio>
```


Il file “portfolio.dtd”

```
<!ELEMENT portfolio (stock)+>
<!ELEMENT stock
          (id, (prezzo | (acquisto, quantita)>
<!ATTLIST stock
          exchange   CDATA     #REQUIRED>
<!ELEMENT id (nome, simbolo) >
<!ELEMENT nome #PCDATA>
<!ELEMENT simbolo #PCDATA>
<!ELEMENT quantita #PCDATA>
<!ELEMENT prezzo #PCDATA>
<!ELEMENT acquisto #PCDATA>
<!ATTLIST prezzo
          unit (“USD”|“DM”|“LST”|“LIT”|“YEN”) “YEN”>
<!ATTLIST acquisto
          unit (“USD”|“DM”|“LST”|“LIT”|“YEN”) “YEN”>
```


Il secondo esempio, più: attributi ID e IDREF

L'elemento dell'indice punta al paragrafo che vi si riferisce:

```
<!DOCTYPE ROMANZO SYSTEM "romanzo.dtd">
<ROMANZO>
  <TITOLO>Tre Uomini in Barca</TITOLO>
  <AUTORE>Jerome K. Jerome</AUTORE> <ANNO>1889</ANNO>
  <CAPITOLO>
    <TITOLO>Capitolo primo</TITOLO>
    <INDICE>
      <EL p="uno">Tre invalidi</EL><EL p="due">Le sofferenze di
      George e Harris</EL><EL p="tre">La vittima di centosette
      malattie inguaribili</EL>
    </INDICE>
    <PARA n="uno">Eravamo in quattro: George, William Samuel
    Harris, io, e Montmorency. Standocene seduti in camera mia,
    fumavamo e parlavamo di quanto fossimo malridotti… <INCISO>
    malridotti, dal punto di vista della salute, <INCISO>intendo,
    </INCISO> naturalmente</INCISO>. </PARA>
    <PARA>Ci sentivamo tutti piuttosto giù di corda,</PARA>
  </CAPITOLO>…
</ROMANZO>
```


Il file “romanzo.dtd”

```
<!ELEMENT ROMANZO   (TITOLO, AUTORE, ANNO, CAPITOLO+)>
<!ELEMENT CAPITOLO  (TITOLO, INDICE, PARA+)>
<!ELEMENT TITOLO    (#PCDATA) >
<!ELEMENT AUTORE    (#PCDATA) >
<!ELEMENT ANNO      (#PCDATA) >
<!ELEMENT INDICE    (EL+) >
<!ELEMENT EL        (#PCDATA) >
<!ATTLIST EL
          p         IDREF          #REQUIRED >
<!ELEMENT PARA      (#PCDATA | INCISO) * >
<!ATTLIST PARA
          n         ID             #IMPLIED >
<!ELEMENT INCISO    (#PCDATA | INCISO)* >
```


Entità

Le entità sono macro testuali a sostituire dei riferimenti brevi, univocamente associati al loro significato esteso.

Ci sono due tipi di entità:

- **Generali**: definite in un DTD, possono essere usate nell'istanza di documento.
- **Parametriche**: definite in un DTD, possono essere usate nello stesso DTD, subito dopo.

Entità generali

```
<!ENTITY Z “pippo”>
```

- Ogni volta che scrivo il nome dell’entità, “&Z;” il processore sostituisce la stringa corrispondente, “pippo”.
- Un’entità della forma “&#NNN;”, dove NNN sono cifre decimali, viene sostituita con il carattere corrispondente nella tabella caratteri di default.

```
<X>Oggi ho visto &Z; </X>
<X>Che bella entit&#224; </X>
```


Complichiamo l'esempio: entità generali

Forniamo supporto per le lettere accentate e forniamo un abbreviazione per i personaggi:

```
<EL p="due">Le sofferenze di &G; e &H;</EL>

<PARA>Eravamo in quattro: &G;, &WSH;, io, e &M;. … </PARA>

<PARA>Ci sentivamo tutti piuttosto gi&ugrave; di
      corda,</PARA>
```

Nel file "romanzo.dtd" metteremo definizioni come:

```
<!ENTITY G       "George">
<!ENTITY H       "Harris">
<!ENTITY WSH     "William Samuel Harris">
<!ENTITY M       "Montmorency">
<!ENTITY ugrave  "&#227">
```


Entità parametriche

```
<!ENTITY % Z “#PCDATA | K”>
```

- Ogni volta che scrivo il nome dell’entità, “%Z;” il processore sostituisce la stringa corrispondente, “#PCDATA | K”.
- Posso usare questa notazione solo all’interno di un DTD

```
<!ELEMENT X (%Z;)*>
<!ELEMENT Y (%Z; | W)* >
```


Complichiamo l’esempio: entità parametriche (I)

Centralizziamo il content model misto, in modo da poterlo cambiare in un colpo solo.

```
<!ENTITY  % testo “(#PCDATA | INCISO) * “>

<!ELEMENT PARA      %testo; >
<!ELEMENT INCISO    %testo; >
```


Complichiamo l'esempio: entità parametriche (II)

Separiamo in un file esterno parte del DTD. Ad esempio, la definizione delle entità carattere:

File entity.dtd

```
 <!ENTITY ugrave  "&#227">
…
```

File romanzo.dtd

```
<!ENTITY  % chars SYSTEM "entity.dtd">
%chars;

Etc....
```


Complichiamo l'esempio: entità parametriche (III)

Realizziamo un DTD compatibile XML e SGML:

- File "romanzo.dtd":

```
<!ELEMENT ROMANZO %min; (TITOLO, AUTORE, ANNO, CAPITOLO+)>
```

- File "romanzosgml.dtd"

```
<!ENTITY % min "- -">
<!ENTITY % dtd SYSTEM
          "romanzo.dtd">
%dtd;
```

- File "romanzoxml.dtd"

```
<!ENTITY % min "">
<!ENTITY % dtd SYSTEM
          "romanzo.dtd">
%dtd:
```


Altre caratteristiche di XML

- Sezioni CDATA (per scrivere testi con caratteri del markup)
- Rigore sintattico (case sensitivity, nessuna minimizzazione)
- Entità predefinite (< >, &, ' ")
- Gestione del white space (e' sempre significativo)
- Attributi riservati (xml-space, xml-lang)
- Marked sections (parti attivabili del DTD)

Sezioni CDATA

- A volte può essere comodo inserire un blocco di caratteri comprendenti anche '&' e '<', senza preoccuparsi di nasconderli dentro ad entità.
- Si usa allora la sezione CDATA, che ha la seguente sintassi:

```
<![CDATA[ dati liberi comprendenti & e < ]]>
```

- L'unica sequenza di caratteri non accettabile è la sequenza `']]>'`, che definisce la fine della sezione CDATA
- Il parser XML passa all'applicazione finale tutti i caratteri che trova fino alla sequenza]]>

```
<para>In HTML, "<![CDATA[ <h1>Questo &egrave; un
titolo</h1>]]>" indica un titolo </para>
```


Altre caratteristiche di XML

- **Case sensitivity**: in XML tutto il markup è case-sensitive (il maiuscolo è diverso dal minuscolo). È quindi necessario usare le maiuscole per ELEMENT, ATTLIST, ecc., e l'elemento <para> è diverso dall'elemento <PARA>.
- **Valori tra virgolette**: tutti i valori di tutti gli attributi debbono avere le virgolette (semplici o doppie, ma in maniera coerente), anche se numeri o appartenenti ad una lista di valori predefiniti.
- **Entità predefinite**: sono pre-definite e non ridefinibili 5 entità:

```
<!ENTITY lt      "&#60;">
<!ENTITY gt      "&#62;">
<!ENTITY amp     "&#38;">
<!ENTITY apos    "'">
<!ENTITY quot    '"'>
```


Il white space

Il white space sono tutti quei caratteri senza aspetto visibile, che influenzano la formattazione del documento.

Essi includono sicuramente lo spazio, il tab, il carriage return ed il line feed, più eventuali altri caratteri.

SGML è nato in ambiente IBM, e le righe erano separate da caratteri chiamati RS e RE (record start e record end). Essi sono poi confluiti in ASCII 13 (CR) e ASCII 10 (LF).

Il white space nel markup

All’interno dei tag, il white space è equivalente ad un singolo spazio, e può essere usato per separare attributi ed altro.

```
<book issue=“3” date=“15/3/97”>
```

è equivalente a

```
<book
  issue   =“3”
  date    =“15/3/97”          >
```


Il white space nel contenuto (1)

Per leggibilità del documento, gli autori possono inserire white space tra elementi e tra contenuto ed elementi.

```
<!ELEMENT autore (nome, cognome, e-mail)>
<!ELEMENT nome    #PCDATA>
<!ELEMENT cognome #PCDATA>
<!ELEMENT e-mail  #PCDATA>
```

```
<autore><nome>Fabio</nome><cognome>Vitali</cogno
me><e-mail>fabio@cs.unibo.it</e-mail></autore>
```

è equivalente a

```
<autore>
    <nome>Fabio</nome>
    <cognome>Vitali</cognome>
    <e-mail>fabio@cs.unibo.it</e-mail>
</autore>
```


Il white space nel contenuto (2)

Nell'esempio precedente, il parser, avendo visto dal DTD che il content model di autore non contiene #PCDATA, può dedurre che il white space non è parte del contenuto.

Tuttavia possono esservi problemi di ambiguità:

```
<para>
    Questo e' un paragrafo con degli
    <emph> spazi </emph>
    che non so come trattare.
</para>
```

Poiché para ha ovviamente un content model misto, non so più distinguere tra white space significativo e white space di presentazione.

Il white space nel contenuto (3)

L’operazione di rimuovere il white space presentazionale lasciando intatto il contenuto vero e proprio è detto ***normalizzazione*** del documento.

Il white space può essere:

- **Preservato**: tutti i caratteri di white space vengono passati intatti all’applicazione
- **Collassato**: tutti i caratteri di white space vengono ridotti ad uno e passati come **un singolo spazio**.
- **Rimosso**: tutti i caratteri di white space vengono cancellati non passati all’applicazione.

XML e il white space

XML adotta convenzioni molto semplici e dirette per il white space:

- ***New line***: Per semplicità ed uniformità, XML trasforma ogni tipo di new line (CRLF, LF e CR) nel solo carattere LF.
- "***If it ain't markup, it's data***": Ogni white space che appare nel contenuto del documento è rilevante, e deve essere passato intatto all'applicazione.
- Tuttavia, un ***parser validante*** è tenuto a precisare all'applicazione quale white space è stato riscontrato in elementi con content model di tipo elemento, cosicché l'applicazione possa decidere cosa farne.

Attributi per white space e lingua

Esistono in XML due attributi riservati (ma da definire se usati):

- **`xml:space`** (valori possibili: “default” o “preserve”) permette all’autore di indicare all’applicazione se è opportuno che mantenga il white space
- **`xml:lang`** (valori possibili: i codici a due lettere di RFC 1766): permette all’applicazione di identificare la lingua in cui è scritto il contenuto di un elemento, per attivare funzionalità dipendenti dalla lingua:
  - Rendering corretto
  - Spell-checking
  - Full-text indexing
  - Editing

Siti Web interessanti

- The XML Cover Pages
  http://www.oasis-open.org/cover/
- The Whirlwind Guide to XML Tools and Vendors
  http://www.infotek.no/sgmltool/guide.htm
- Free XML tools and software
  http://www/xmlsoftware.org/
  http://www.garshol.priv.no/download/xmltools/
- W3C XML page
  http://www.w3.org/XML/

Riferimenti (1)

- XML
  - J. Bosak, ***XML, Java, and the future of the Web***, **`http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm`**
  - T. Bray, J. Paoli, C.M. Sperberg-McQueen, ***Extensible Markup Language (XML) 1.0***, W3C Recommendation, 10 February 1998, **`http://www.w3.org/TR/REC-xml`**
  - T. Bray, The annotated XML Specification,1998, **`http://www.xml.com/axml/testaxml.htm`**