La sintassi di SGML ∩ XML

Fabio Vitali

17 gennaio 2000

Introduzione

Qui esaminiamo in breve tutti gli aspetti di SGML che sono in comune con XML:

- I componenti di un documento
- La struttura di un documento
- La sintassi dei DTD
- Le marked sections

In lezioni successive guarderemo a tutto quello di SGML che non è transitato in XML e viceversa.

I componenti di SGML

Un documento SGML contiene una varietà dei seguenti componenti

- Elementi
- Attributi
- Entità
- Testo (o #PCDATA)
- Commenti

Elementi

- 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>
```


Attributi

- 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>
```


Entità

- 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!”
```


Testo

- 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 SGML usa questo nome per indicare il contenuto di elementi di testo.

Commenti

I documenti SGML 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 SGML 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 SGML -->
```


I documenti SGML

Un documento in un linguaggio di markup definito sulla base di SGML è sempre composto delle seguenti tre parti:

- Dichiarazione SGML
- DTD
- Istanza del documento

SGML Declaration

```
<!SGML “ISO 8879:1986” car. spec.>
```

- La dichiarazione SGML contiene le istruzioni di partenza delle applicazioni SGML.
- Essa permette di specificare valori fondamentali come la lunghezza dei nomi degli elementi, il set di caratteri usati, le specifiche caratteristiche di minimizzazione ammesse, ecc.)
- Non è obbligatoria. Se è assente, viene usata una dichiarazione di default detta “***Reference Concrete Syntax***”.

Document Type Declaration

```
<!DOCTYPE nome TIPO [markup] >
```

- 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 SGML, si parla di modellare una classe (cioè una collezione omogenea) di documenti attribuendogli un tipo.

La Document Instance

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

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

Sintassi dei DTD

- Una precisazione
- <!DOCTYPE … >
- <!ELEMENT … >
- <!ATTLIST … >
- <!ENTITY … >: Entità generali
- <!ENTITY % … >: Entità parametriche
- <!NOTATION … >
- <? … > - Istruzioni di processo
- Marked sections
- SGML declaration: a volo d'angelo

Una precisazione

- I DTD XML e SGML sono molto simili. A parte minime modifiche, ogni DTD XML è anche un DTD SGML.
- A noi interessa soprattutto fare documenti SGML che possano passare per documenti XML, e viceversa.
- Quindi esponiamo come sintassi del DTD solo quella comune a entrambi i linguaggi. Tocchiamo solo in parte la sintassi di SGML che non è stata ereditata da XML.
- Se facciamo degli esempi con il resto della sintassi SGML, è solo per approfondimento e solo brevemente.

La dichiarazione di tipo

- Il <!DOCTYPE ... > è la dichiarazione del tipo di documento. Essa permette alle applicazioni SGML 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 …>

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

- ST & ET: minimizzazione del tag iniziale (ST) e finale (ET): può assumere i valori ‘-’ (***obbligatorio***) o ‘o’ (***omissibile***). Per noi sono sempre obbligatori.
- Content-model: la specificazione formale del contenuto permesso nell’elemento, secondo una sintassi specifica di gruppi di modelli.

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.
- Un content model (CM) è 'ANY', 'EMPTY', oppure un gruppo di CM più elementari.
- Un gruppo di CM è sempre circondato da parentesi. Può contenere la specifica #PCDATA, la specifica di un elemento SGML, o di un altro gruppo di CM più elementare. Ogni specifica è separata da un separatore. Alla fine ci può essere un operatore di ripetizione.

ANY, EMPTY, #PCDATA

- ANY: significa che qualunque content model è ammesso. Ogni elemento definito nel DTD può comparire qui dentro in qualunque ordine e numero.
- EMPTY: Questo è un elemento vuoto, o senza contenuto. In questo caso nel documento esso appare come tag semplice, senza tag finale.

  Def.: `<!ELEMENT HR - - EMPTY>`
  Uso: “<HR>”
- #PCDATA: (Parsed Character Data): il contenuto testuale del documento. Include caratteri ed entità generali. È naturalmente in numero multiplo.

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.

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***.

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***.

Element content semplice

```
<!ELEMENT sezione - - (titolo, abstract, para) >
```

Un elemento contiene solo altri elementi, senza parti opzionali.

Dentro all'elemento sezione ci deve essere un titolo, seguito da un abstract, seguito da un para.

```
<sezione>
      <titolo> … </titolo>
      <abstract> … </abstract>
      <para> … </para>
</sezione>
```


Element content con elementi facoltativi o ripetuti

```
<!ELEMENT sezione - - (titolo, abstract?, para+)>
```

Un elemento contiene solo altri elementi, ma alcuni possono essere opzionali e altre in presenza multipla.

Dentro all'elemento sezione ci deve essere un titolo, seguito facoltativamente da un abstract, seguito da almeno (ma anche più di) un para.

```
<sezione>
      <titolo> … </titolo>
      <para> … </para>
      <para> … </para>
</sezione>
```


Element content complesso

```
<!ELEMENT sezione - - (titolo, (abstract | para)+) >
```

Un elemento contiene solo altri elementi, ma gli operatori di ripetizione e i separatori permettono sequenze complesse di elementi.

Dentro all'elemento sezione ci deve essere un titolo, seguito da almeno uno di abstract o para, che poi possono ripetersi in qualunque ordine e in qualunque numero.

```
<sezione>
      <titolo> … </titolo>
      <para> … </para>
      <abstract> … </abstract>
      <para> … </para>
</sezione>
```


Character content

```
<!ELEMENT para - - #PCDATA >
```

Un elemento contiene soltanto caratteri stampabili e entità. Nessun altro elemento è ammesso all'interno.

```
<para>Questo &egrave; lecito</para>
```


Mixed content

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

Un elemento contiene sia caratteri stampabili ed entità, sia altri elementi.

```
<para>Questo &egrave; un paragrafo lecito con
alcune <bold> parole in grassetto </bold> e poi
<bold> ancora altre </bold>. </para>
```

Nota: in XML la specifica #PCDATA deve sempre essere la prima di qualunque gruppo di mixed content. In SGML non è obbligatorio, ma è buono stile.

Lista di attributi: <!ATTLIST … >

La dichiarazione <!ATTLIST… > permette di definire una lista di attributi lecita ad un elemento SGML 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
        …
>
```


Attributo di tipo stringa

```
<!ATTLIST doc
      linguaggio     CDATA     “HTML” >
```

CDATA significa “character data”, e indica qualunque sequenza di caratteri (tranne “<“ e le virgolette già usate come delimitatore), ma non entità o elementi.

Attributi di tipo lista

```
<!ATTLIST doc
   stato      (bozza | impaginato | finale) “bozza”
>
```

Solo uno dei valori elencati nella lista può essere accettato. Ogni altro valore genererà un errore.

Da notare che in SGML (ma non XML), ogni valore possibile in una lista di attributi dello stesso elemento deve essere unico. La definizione seguente genera un errore:

```
<!ATTLIST doc
   stato      (bozza | impaginato | finale) “bozza”
   posizione  (iniziale | mediana | finale) “iniziale”
>
```


Attributi di tipo predefinito

- `ENTITY` o `ENTITIES`: una o una lista di entità predefinite. Posso specificarne il nome (senza né & né ;) come valore.
- `ID`: un identificativo univoco all'interno del documento.
- `IDREF` o `IDREFS`: un riferimento (o una lista di riferimenti) ad un identificativo definito altrove nel documento con un attributo ID. L'ID corrispondente deve esistere.
- `NMTOKEN` o `NMTOKENS`: un nome (o una lista di nomi). Un nome è una stringa di caratteri alfanumerici che include le lettere maiuscole e minuscole, i numeri e i caratteri ".", "-", "_", ":", ma non spazi, altri segni di punteggiatura, e altri caratteri particolari.

Valore di default letterale

L’attributo assume quel valore se non ne viene specificato un altro. Ad esempio, in

```
<!ATTLIST doc
      stato (bozza | impaginato | finale) “bozza”
>
```

l’uso

```
<doc stato=“bozza”> Questo &egrave; un documento </doc>
```

e l’uso

```
<doc> Questo &egrave; un documento </doc>
```

sono uguali.

Valore di default #FIXED

L'attributo assume automaticamente quel valore e non ne può assumere un altro. Il tentativo di assegnare un altro valore a quell'attributo produce un errore:

Ad esempio, in

```
<!ATTLIST doc
      stato        CDATA        #FIXED "bozza" >
```

l'uso

```
<doc stato="finale"> Questo &egrave; un documento
</doc>
```

è un errore.

Valore di default #REQUIRED

Non esiste valore di default: l'autore deve fornire ogni volta un valore.

Ad esempio, in

```
<!ATTLIST doc
   stato (bozza | impaginato | finale)#REQUIRED
>
```

l'uso

```
<doc> Questo &egrave; un documento </doc>
```

è un errore.

Valore di default `#IMPLIED`

Non è obbligatorio specificare un valore per questo attributo. Se esiste, verrà considerato il valore fornito, altrimenti l'applicazione deve fornirne un valore proprio.

Gli attributi di tipo ID debbono avere un valore `#REQUIRED` (l'autore deve fornire un identificativo univoco ogni volta) o `#IMPLIED` (l'applicazione si preoccupa di fornire un identificativo interno).

Entità generali: <!ENTITY … >

Le entità generali sono elementi di contenuto definite nel DTD e richiamate nel documento.

Durante la lettura del documento i richiami delle entità vengono sostituite con il valore definito (espansione dell'entità)

Le entità generali sono permesse nel contenuto degli elementi e nei valori degli attributi.

Entità generali: definizione e uso

Definizione: `<!ENTITY nome valore>`
Uso: `&nome;`

Definizione (nel DTD):

```
<!ENTITY xml “Extensible Markup Language”>
<!ENTITY dataxml “5/3/2000”>
```

Uso (nel documento):

```
<para> <data n=“&dataxml;”>Presto</data>
impariamo l’&xml; </para>
```

Espansione:

```
<para> <data n=“5/3/2000”>Presto</data>
impariamo l’Extensible Markup Language </para>
```


Entità parametriche: <!ENTITY % … >

- Le entità parametriche sono elementi di specifica definiti nel DTD e usati nel DTD stesso, dopo la loro definizione.
- Servono per raccogliere in un'unica locazione definizioni di content model, o di attributi, o di valori comuni a molti elementi.
- Durante la lettura del DTD le entità parametriche vengono sostituite con il loro valore e questo viene usato per la definizione degli elementi.
- Invece del carattere '&' viene usato (anche nella definizione!!!) il carattere '%'.

Entità parametriche: definizione e uso

Definizione: `<!ENTITY % nome valore>`
Uso: `%nome;`

Definizione (nel DTD):

```
<!ENTITY % inline “(#PCDATA | bold)*”>
```

Uso (nel DTD):

```
<!ELEMENT para1 %inline;>
<!ELEMENT para2 (%inline; | italic)*>
```

Espansione:

```
<!ELEMENT para1 (#PCDATA | bold)*>
<!ELEMENT para2 ((#PCDATA | bold)* | italic)*>
```


Entità esterne

Non è necessario che il valore di espansione dell'entità sia specificato nella definizione.

Per modularità, condivisione, o dimensioni eccessive, può essere comodo inserire il valore dell'entità in un file a parte, e definire l'entità come esterna usando la keyword `SYSTEM`.

```
<!ENTITY % blocco-dtd SYSTEM “blocco.dtd”>
<!ENTITY % grosso-testo SYSTEM “testo.sgm”>
```


Notazioni particolari: <!NOTATION … >

Quando si vogliono utilizzare delle informazioni non-SGML all'interno di un documento SGML, è opportuno utilizzare la specifica di notazione.

Tramite la specifica `<!NOTATION … >` si indica alle applicazioni SGML che esistono altre notazioni oltre a SGML nel documento.

Le parti del documento nella notazione specificata vengono o inserite in un apposito elemento in cui è definito un attributo con la keyword `NOTATION`

Alternativamente, è possibile indicare una entità esterna in una notazione particolare.

Notazioni: definizione e uso

Definizione:

```
<!NOTATION gif SYSTEM “gif.doc”>
<!NOTATION jpeg SYSTEM “jpeg.doc”>
<!ELEMENT immagine #PCDATA>
<!ATTLIST immagine
        tipo NOTATION(gif|jpeg) #REQUIRED>
<!ENTITY img SYSTEM “file.gif” NDATA gif>
```

Uso:

```
<immagine tipo=“gif”>GIF89/1 AZ5TYTR…
</immagine>
<para>e poi metto qui quest’immagine: &img;
</para>
```


Istruzioni di processo: <? ... >

- 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 SGML 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).
- In questi casi, l'applicazione può inserire elementi di tipo PI: `<?NEWPAGE>` . Da notare che in XML la sintassi richiede un altro '?' alla fine: `<?NEWPAGE?>`

Marked sections

- A volte si vorrebbe in un DTD specificare due content model incompatibili per lo stesso elemento, da attivare o disattivare in casi diversi.
- Ad esempio, si vogliono permettere degli elementi ‘commento’ in un documento di diffusione interna, e non permetterli in un documento pubblicato.
- In questo caso si definiscono delle sezioni condizionali (in SGML: marked sections; in XML: conditional sections) che si esaminano o si ignorano secondo quanto precisato.
- Nella sintassi delle marked sections, le parti indicate all’interno di `<![INCLUDE[ dati ]]>` vengono considerate, mentre le parti indicate all’interno di `<![IGNORE[ dati ]]>` vengono ignorate.

Marked sections: uso

```
<!ENTITY % usointerno “INCLUDE”>
<!ENTITY % pubblicazione “IGNORE”>

<![%usointerno;[
      <!ELEMENT libro (commento | capitolo)+ >
]]>
<![%pubblicazione;[
      <!ELEMENT libro (capitolo)+ >
]]>
```

Cambiando il valore delle due entità parametriche si possono avere DTD anche molto diversi sulla base dello stesso documento.

Identificatori SYSTEM e PUBLIC

In molti casi nella specifica del DTD o della dichiarazione SGML, è possibile specificare dei valori usando le keyword PUBLIC o SYSTEM

- SYSTEM fa riferimento ad un file contenuto sul sistema che fornisce l'applicazione SGML. A seguito della dichiarazione di SYSTEM, ci deve essere il nome del file, secondo la sintassi del sistema operativo usato.
- PUBLIC fa riferimento a nomi logici contenuti in un apposito registro posto sul sistema. Questo contiene identificatori pubblici dell'ISO, formalmente registrati, oppure non registrati o informali.
  Un identificatore pubblico è diviso in 3 campi separati da '//': tipo di identificatore e proprietario, nome identificativo, lingua. Ad esempio:
  - `ISO 8879:1986//ENTITIES Publishing//EN`
  - `+//ISBN 0-933186::IBM// …`
  - `-//DEC//ENTITIES Tech Chars//EN`

Conclusioni

Qui abbiamo parlato delle caratteristiche di SGML in comune con XML:

- La struttura dei documenti
- Elementi, entità, attributi, notazioni
- Come descrivere il content model di un elemento

Riferimenti

Wilde's WWW, capitolo 4

Altri testi:

- E. Maler, J. Al Andaloussi, ***Developing SGML DTDs, from text to model to markup***, Prentice Hall, 1997