

Esempi di SGML \cap XML

Fabio Vitali



Introduzione

Qui esaminiamo in breve degli esempi delle varie caratteristiche di SGML e XML.

- ◆ Elementi
- ◆ Attributi
- ◆ Entità generali
- ◆ Entità parametriche
- ◆ Marked sections



Esempi di sviluppo del DTD

Ricapitolando, ci sono quattro caratteristiche di SGML che vale la pena considerare a fondo:

- ◆ Elementi `<A> ... `
- ◆ Attributi ` ... `
- ◆ Entità: `<P>In HTML, si scrive entità</P>`
- ◆ Marked sections



Elementi

Il content model di un elemento definisce quali e quanti elementi possono essere contenuti in un elemento.

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.

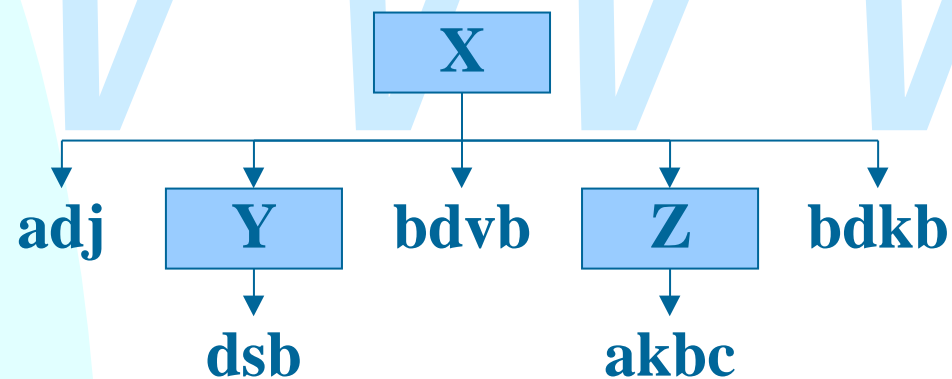


Content model qualunque

<!ELEMENT X - - ANY>

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

<X>adj<Y>dsb</Y>bdvb<Z>akbc</Z>bdkb</X>



Content model vuoto

<!ELEMENT X - O EMPTY>

- ◆ L'elemento X non può contenere niente. E' equivalente scrivere il solo tag iniziale, o tag iniziale e finale (per questo se il content model è EMPTY, il secondo tag deve essere segnato come omissibile).

<X></X>

<X>

- ◆ N.B. In XML il tag vuoto ha una sintassi diversa:

<X></X>

<X/>

X

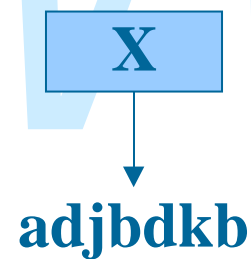


Content model testo

<!ELEMENT X - - #PCDATA>

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

<X>adjbdkb</X>

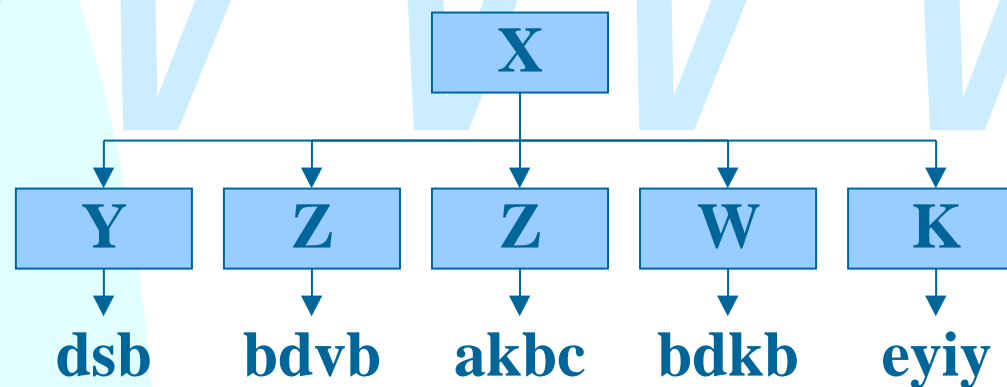


Content model strutturato

<!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.

<X><Y>dsb</Y><Z>bdvb<Z><Z>akbc</Z><W>bdkb</W><K>eyiy</K></X>

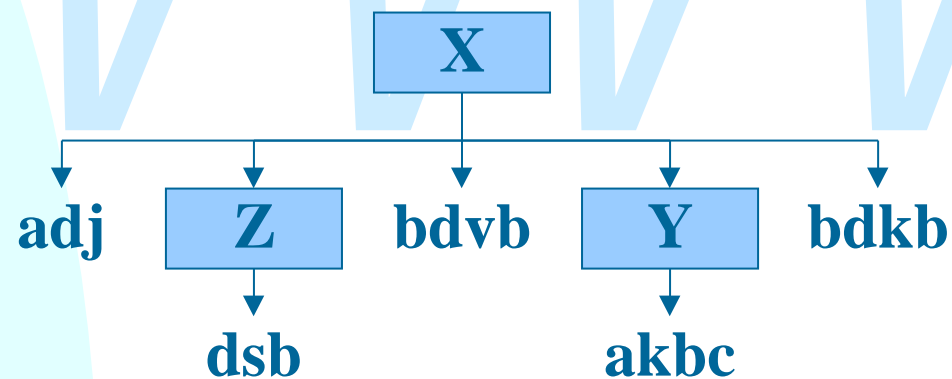


Content model misto

<!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.

<X>adj<Z>dsb</Z>bdvb<Y>akbc</Y>bdkb</X>



Separatori e ripetizioni

I separatori in SGML e XML sono due:

- ◆ La virgola indica elementi obbligatori e nel loro ordine.
- ◆ La barra verticale indica elementi alternativi

Gli operatori di ripetizione sono quattro

- ◆ Niente: elemento obbligatorio e non ripetibile
- ◆ ?: elemento facoltativo e non ripetibile
- ◆ +: elemento obbligatorio e ripetibile
- ◆ *: elemento facoltativo e ripetibile



Un esempio semplice

Una lettera commerciale è composta da un destinatario, un corpo e dei saluti. Il destinatario è composto di nome e ufficio interno. Il corpo è composto di paragrafi. I saluti sono testo semplice.

```
<!DOCTYPE lettera SYSTEM "lettera.dtd">
<lettera>
  <destinatario>
    <nome>Mario Rossi</nome>
    <ufficio>Ufficio Acquisti</ufficio>
  </destinatario>
  <corpo>
    <para> Comprate tutto! </para>
  </corpo>
  <saluti> Arrivederci </saluti>
</lettera>
```



II DTD relativo

```
<!ELEMENT lettera - - (destinatario, corpo, saluti)>
```

```
<!ELEMENTI destinatario - - (nome, ufficio) >
```

```
<!ELEMENT corpo - - (para)+>
```

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

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



Complichiamo l'esempio: alternative

Se il destinatario è interno, è composto da nome e ufficio interno. Altrimenti è composto da nome, ditta, indirizzo e città.

```
<!DOCTYPE lettera2 SYSTEM "lettera2.dtd">
<lettera2>
  <destinatario>
    <nome>Mario Rossi</nome>
    <ditta>ACME, Inc.</ditta>
    <indirizzo>Via Garibaldi 1</indirizzo>
    <citta>Bologna</citta>
  </destinatario>
  ...
</lettera2>
```



Il DTD relativo a lettera2

```
<!ELEMENT lettera2 - - (destinatario, corpo, saluti)>
<!ELEMENTI destinatario - -
    (nome, (ufficio | (ditta, indirizzo, citta)))>
<!ELEMENT corpo - - (para)+>
<!ELEMENT saluti - - #PCDATA>
<!ELEMENT para - - #PCDATA>
<!ELEMENT ditta - - #PCDATA>
<!ELEMENT indirizzo - - #PCDATA>
<!ELEMENT citta - - #PCDATA>
```

N.B.: le parentesi servono per evitare ambiguità. Allo stesso motivo “nome”, esistente in entrambe le possibilità, è stato estratto per evitare ambiguità di parsing.



Complichiamo l'esempio: ripetizioni

L'indirizzo è composto da una o più righe.

```
<!DOCTYPE lettera3 SYSTEM "lettera3.dtd">
<lettera2>
  <destinatario>
    <nome>Mario Rossi</nome>
    <ditta>ACME, Inc.</ditta>
    <indirizzo>
      <riga>c/o Associated Producers</riga>
      <riga>Via Rossi 1</riga>
      <riga>Casteldebole</riga>
    </indirizzo>
    <citta>Bologna</citta>
  </destinatario>
  ...
</lettera3>
```



Il DTD relativo a lettera3

```
<!ELEMENT lettera3 - - (destinatario, corpo, saluti)>
<!ELEMENTI destinatario - -
    (nome, (ufficio | (ditta, indirizzo, citta)))>
<!ELEMENT corpo - - (para)+>
<!ELEMENT saluti - - #PCDATA>
<!ELEMENT para - - #PCDATA>
<!ELEMENT ditta - - #PCDATA>
<!ELEMENT indirizzo - - (riga)+>
<!ELEMENT citta - - #PCDATA>
<!ELEMENT riga - - #PCDATA>
```



Complichiamo l'esempio: content model misto

I paragrafi possono contenere anche corsivi e grassetti.

```
<!DOCTYPE lettera4 SYSTEM "lettera4.dtd">
<lettera4>
  <destinatario>
    ...
  </destinatario>
  <corpo>
    <para> Comprate <b>tutto!</b> </para>
  </corpo>
  <saluti> Arrivederci </saluti>
</lettera4>
```



Il DTD relativo a lettera4

```
<!ELEMENT lettera4 - - (destinatario, corpo, saluti)>
<!ELEMENTI destinatario - -
    (nome, (ufficio | (ditta, indirizzo, citta)))>
<!ELEMENT corpo - - (para)+>
<!ELEMENT saluti - - #PCDATA>
<!ELEMENT para - - (#PCDATA | bold | italic)*>
<!ELEMENT bold - - (#PCDATA | italic)*>
<!ELEMENT italic - - (#PCDATA | bold)*>
<!ELEMENT ditta - - #PCDATA>
<!ELEMENT indirizzo - - (riga)+>
<!ELEMENT citta - - #PCDATA>
<!ELEMENT riga - - #PCDATA>
```



Attributi

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 tre 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**



Attributi stringa

```
<!ATTLIST X
```

```
  a
```

```
  CDATA
```

```
  "pippo"
```

```
>
```

- ◆ Il tag iniziale di X può contenere un attributo chiamato "a". Qualunque stringa è lecita. Se non viene specificata una stringa, il valore definito per default è "pippo"
- ◆ `<X a="pluto">adj</X>`
- ◆ `<X>bfg</X>` `<!-- a vale "pippo" -->`



Attributi lista

```
<!ATTLIST X  
  a (pippo|pluto|paperino) "pippo"  
>
```

- ◆ Il tag iniziale di X può contenere un attributo chiamato "a". Sono leciti solo i valori descritti. Se non viene specificata una stringa, il valore definito per default è "pippo"
- ◆ `<X a="pluto">adj</X>`
- ◆ `<X>bfg</X>` `<!-- a vale "pippo" -->`
- ◆ `<X a="paperino">ljdn</X>`
- ◆ `<X a="topolino">kdjbk</X>` `<!-- errore -->`



Attributi ID

<!ATTLIST X

a

ID

#REQUIRED

>

- ◆ Il tag iniziale di X può contenere un attributo chiamato “a”. Sono leciti solo valori unici su tutto il documento. L’elemento X assume un’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.
- ◆ `<X a="pluto">adj</X><X a="pippo">bfg</X>`
- ◆ `<X a="pluto">adj</X><X a="pluto">bfg</X>`
`<!-- errore -->`



Attributi IDREF

```
<!ATTLIST X
```

```
  a
```

```
  ID
```

```
  #REQUIRED
```

```
  b
```

```
  IDREF
```

```
  #IMPLIED
```

```
>
```

- ◆ Il tag iniziale di X può contenere un attributo chiamato “a” ed uno chiamato “b”. I valori di “a” debbono essere unici. I valori di “b” debbono essere uguali ad un valore di “a” esistente da qualche parte nel documento.

- ◆ `<X a="pluto">adj</X><X b="pluto">bfg</X>`

- ◆ `<X a="pluto">adj</X><X b="pippo">bfg</X>`
`<!-- errore -->`



Valori di default negli attributi

I valori di default importanti in SGML e 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.



Complichiamo l'esempio: attributi testo e lista

Ogni lettera deve avere una data di creazione e può avere un autore, che non fanno parte del contenuto della lettera. Inoltre è necessario specificare se il destinatario è interno o esterno.

```
<!DOCTYPE lettera5 SYSTEM "lettera5.dtd">
<lettera5 data="28/01/2000" autore="FV">
  <destinatario tipo="esterno">
    ...
  </destinatario>
  <corpo>
    ...
  </corpo>
  <saluti> ... </saluti>
</lettera5>
```



Il DTD relativo a lettera5

```
<!ELEMENT lettera5 - - (destinatario, corpo, saluti)>
<!ATTLIST lettera5
    data          CDATA          #REQUIRED
    autore        CDATA          #IMPLIED
>
<!ELEMENTI destinatario - -
    (nome, (ufficio | (ditta, indirizzo, citta)))>
<!ATTLIST destinatario
    tipo (interno | esterno)      "interno"
>
<!ELEMENT corpo - - (para)+>
<!ELEMENT saluti - - #PCDATA>
...
```



Complichiamo l'esempio: attributi ID e IDREF

Ogni paragrafo deve avere un identificativo univoco. L'elemento "link", che può essere contenuto nei paragrafi, può creare un link ad un paragrafo precedente.

```
<!DOCTYPE lettera6 SYSTEM "lettera6.dtd">
<lettera6 data="28/01/2000" autore="FV">
  ...
  <corpo>
    <para id="P1"> Comprate tutto! </para>
    <para> Imparate il
      <link v="P1">paragrafo precedente</link>
    </para>
  </corpo>
  <saluti> ... </saluti>
</lettera6>
```



Il DTD relativo a lettera6

```
<!ELEMENT lettera6 - - (destinatario, corpo, saluti)>
<!ELEMENT corpo - - (para)+>
<!ELEMENT para - - (#PCDATA | bold | italic | link)*>
<!ATTLIST para
    id ID #IMPLIED
>
<!ELEMENT link - - (#PCDATA | bold | italic)*>
<!ATTLIST link
    v IDREF #REQUIRED
>
..
```



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>
```



Entità parametriche

```
<!ENTITY % Z "#CDATA | Z">
```

- ◆ Ogni volta che scrivo il nome dell'entità, "%Z;" il processore sostituisce la stringa corrispondente, "#CDATA | Y".
- ◆ Posso usare questa notazione solo all'interno di un DTD

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

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



Complichiamo l'esempio: entità generali

Forniamo supporto per le lettere accentate e forniamo due o tre forme standard di saluto

```
<!DOCTYPE lettera7 SYSTEM "lettera7.dtd">
<lettera7 data="28/01/2000" autore="FV">
  ...
  <corpo>
    <para> Ol&agrave;! Comprate tutto! </para>
    ...
  </corpo>
  <saluti> &CS; </saluti>
</lettera7>
```

In fase di elaborazione, "agrave" diventerà "à" e "CS" diventerà "Cordiali saluti".



Il DTD relativo a lettera7

```
<!ELEMENT lettera7 - - (destinatario, corpo, saluti)>
<!ELEMENT corpo - - (para)+>
<!ELEMENT para - - (#PCDATA | bold | italic | link)*>
...
<!ENTITY agrave "&#224;" >
<!ENTITY CS "Cordiali saluti">
<!ENTITY AP "A presto">
...
```



Complichiamo l'esempio: entità parametriche (I)

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

```
<!DOCTYPE lettera8 SYSTEM "lettera8.dtd">
<lettera8 data="28/01/2000" autore="FV">
...
<corpo>
  <para> Ol&agrave;! Comprate tutto! </para>
  ...
</corpo>
  <saluti> &CS; </saluti>
</lettera8>
```

(il documento è identico!!!)



Il DTD relativo a lettera8

```
<!ELEMENT lettera8 - - (destinatario, corpo, saluti)>
<!ELEMENT corpo - - (para)+>
...
<!ENTITY % inline "(#PCDATA | bold | italic | link)*">
<!ELEMENT para - - %inline;>
<!ELEMENT link - - %inline;>
<!ELEMENT bold - - %inline;>
<!ELEMENT italic - - %inline;>
...
```

- Cosa abbiamo perso?



Complichiamo l'esempio: entità parametriche (II)

- Raccogliamo e separiamo tutte le entità per le lettere strane, perché sono troppe e confondono le idee.
- Inoltre rendiamo il frammento di DTD compatibile sia con SGML che con XML (XML non ha le specifiche di minimizzazione!!!)

```
<!DOCTYPE lettera9 SYSTEM "lettera9sgml.dtd">
<lettera9 data="28/01/2000" autore="FV">
...
<corpo>
  <para> Ol&agrave;! Comprate tutto! </para>
...
</corpo>
  <saluti> &CS; </saluti>
</lettera9>
```



II DTD relativo a lettera9

- File “entities.dtd”:

```
<!ENTITY agrave “&#224;” >
```

```
<!ENTITY egrave “&#232;” >
```

- File “lettera9.dtd”:

```
<!ENTITY % entities SYSTEM “entities.dtd”>
```

```
%entities;
```

```
<!ELEMENT lettera9 %min; (destinatario, corpo, saluti)>
```

```
<!ELEMENT corpo %min; (para)+>
```

- File “lettera9sgml.dtd”

```
<!ENTITY % min “- -”>
```

```
<!ENTITY % dtd SYSTEM  
    “lettera9.dtd”>
```

```
%lettera9.dtd
```

- File “lettera9xml.dtd”

```
<!ENTITY % min “”>
```

```
<!ENTITY % dtd SYSTEM  
    “lettera9.dtd”>
```

```
%lettera9.dtd
```



Marked sections

```
<![INCLUDE[
    Roba da includere
]]>
<![EXCLUDE[
    Roba da escludere
]]>
```

- ◆ Permette di specificare parti del DTD che vengono considerate ed escluse in alternativa.
- ◆ Per esempio permettono di specificare due content model per lo stesso elemento, da attivare in alternativa.



Complichiamo l'esempio: Marked Section

Creiamo due DTD, uno in cui certi elementi sono leciti, e l'altro in cui non lo sono. Questo deve essere fatto con un unico file, per gestire insieme tutte le evoluzioni successive.

```
<!DOCTYPE lettera10 SYSTEM "lettera10.dtd">
<lettera10 data="28/01/2000" autore="FV">
...
<corpo>
  <para> Comprate <bum>tutto!</bum></para>
  <para> Imparate il
    <link v="P1">paragrafo precedente</link>
  </para>
</corpo>
<saluti> &CS; </saluti>
</lettera10>
```



Il DTD relativo a lettera10

- File “lettera10.dtd”:

```
<!ELEMENT lettera10 - - (destinatario, corpo, saluti)>
<!ELEMENT corpo - - (para)+>
<![%BUM;[
  <!ENTITY % inline "(#PCDATA | italic | link | bum)*">
]]> <![%NoBUM; [
  <!ENTITY % inline "(#PCDATA | italic | link )*">
]]>
<!ELEMENT para - - %inline;>
```

- File “lettera10-bum.dtd”

```
<!ENTITY % BUM "EXCLUDE">
<!ENTITY % NoBUM "INCLUDE">
<!ENTITY % dtd SYSTEM
  "lettera10.dtd">
%lettera10.dtd
```

- File “lettera10+bum.dtd”

```
<!ENTITY % BUM "INCLUDE">
<!ENTITY % NoBUM "EXCLUDE">
<!ENTITY % dtd SYSTEM
  "lettera10.dtd">
%lettera10.dtd
```



Conclusioni

Qui abbiamo fatto degli esempi di DTD XML e SGML.

- ◆ Elementi `<A> ... `
- ◆ Attributi ` ... `
- ◆ Entità: `<P>In HTML, si scrive entità</P>`
- ◆ Marked sections



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

