# Introduzione al DTD

Mario Arrigoni Neri

# XML ben formati ed XML validi

- I due diversi livelli di standardizzazione generano due livelli di “correttezza”:
  - XML ben formato: un file XML è ben formato quando obbedisce a tutte le regole di XML, ad esempio deve avere il nesting corretto dei tag, un unico elemento radice e non deve avere errori “di sintassi”
  - XML valido: per essere valido un XML deve essere ben formato e, inoltre, deve presentare i tag corretti per la specifica applicazione, nel giusto ordine e con i giusti attributi
- Es: <title><book/><book> Titolo </book></title> è ben formato ma non è tipicamente valido
- Per valutare la buona formazione basta conoscere le regole sintattiche di XML
- La validazione dipende dalla specifica applicazione e cioè dallo specifico linguaggio

# DTD

- DTD (Document Type Definition) fa parte dello standard XML e permette di specificare le regole di validazione del particolare linguaggio
- Il cuore di una applicazione XML è il parser, ovvero quel modulo che legge il documento e ne crea una rappresentazione interna adatta all'elaborazione
- Un parser validante, in presenza di un DTD, è in grado di verificare la validità del documento
- Un parser non validante, invece, anche in presenza di un DTD può solo controllare la buona forma
- Visto dal DTD un file XML è composto da elementi, tag, attributi, entità, PCDATA e CDATA

# <!DOCTYPE>

- Nel caso del DTD il validatore è identificato come applicazione DOCTYPE
- Il tag contiene il riferimento al DTD, che può essere sia esterno che interno al file XML.
  - DTD interno (o in-line): <!DOCTYPE tipodoc [.. Codice DTD ..]>
  - DTD esterno: il tag contiene il riferimento al file DTD
    <!DOCTYPE tipodoc SYSTEM “url del file”>
- Esempio:

```
<?xml version=“1.0”?>
<!DOCTYPE note SYSTEM “note.dtd”>
<note>…
</note>
```

Il tipo del documento è uguale all’elemento radice

# elementi

- Il DTD definisce i tipi di elementi che possono essere presenti nel file xml
- Per ogni tipo di elemento viene indicato il tipo di contenuto, che può essere:
  - Any content: indica che ogni contenuto è ammissibile.
    Es: <!ELEMENT memo ANY>
  - Empty content: un elemento vuoto non può contenere alcun testo tra il tag di inizio e quello di chiusura e può quindi essere rappresentato da un tag vuoto. Es: <!ELEMENT br EMPTY>
  - Simple content: è un elemento il cui contenuto è composto da testo. In questo caso #PCDATA è acronimo di “Parsed Character Data”.
    Es: <!ELEMENT message (#PCDATA)>
  - Element content: è il caso tipico in cui il contenuto è composto da sottoelementi.
    Es: <!ELEMENT note (to, from, title, message)>
  - Mixed content: sono elementi che contengono testo misto ad altri elementi

# element content - sequenza

- Un elemento composto da una sequenza deve contenere una lista dei sottoelementi elencati, con un elemento per ogni tipo e nello stesso ordine
- I singoli sottoelementi sono divisi da virgole

| DTD | XML valido |
| --- | --- |
| <!ELEMENT note (to, from, title, message)> | <note><br><to/><from/><br><title/><message/><br></note> |

# element content - alternativa

- Indica che il contenuto deve essere uno tra quelli elencati
- I singoli sottoelementi sono divisi da barre verticali (l'or bitwise del C)

| DTD | XML valido |
| --- | --- |
| <!ELEMENT a (b \| c \| d) > | <a><br><br></a> |

# element content - ripetizione

- E’ possibile definire in vari modi le cardinalità dei sottoelementi:
  - Nessuna specifica: l’argomento deve comparire esattamente una volta
    <!ELEMENT a (b)>  è permesso solo <a><b/></a>
  - ?: opzione. Specifica che il sottoelemento è facoltativo
    <!ELEMENT a (b?)>  sono leciti sia <a/> che <a><b/></a>
  - +: ripetizione. Il sottoelemento deve figurare una o più volte
    <!ELEMENT a (b+)>  <a><b/></a>, <a><b/><b/></a>, ecc...
  - *: ripetizione. Il sottoelemento può figurare un numero qualsiasi di volte, o anche non figurare affatto
    <!ELEMENT a (b*)>  <a/>, <a><b/></a>, <a><b/><b/></a>, ecc...

# element content complessi

- I costrutti possono combinarsi dando origine ad espressioni regolari
  - <!ELEMENT sezione (titolo, abstract?, para+)>
    ogni sezione ha un titolo, può avere un abstract opzionale, seguito da almeno un paragrafo
  - <!ELEMENT sezione (titolo, (abstract | para)+)>
    Dentro all'elemento sezione ci deve essere un titolo, seguito da almeno un abstract o un para, che poi possono ripetersi in qualunque ordine e numero
  - <!ELEMENT sezione (titolo, abstract*, para+)>
    Ogni elemento sezione è composto da un titolo, da una sequenza opzionale di abstract e da una sequenza si para composta da almeno un para
  - <!ELEMENT sezione (titolo, (sottotitolo | abstract)?, para+)>
    Ogni sezione è data da un titolo, da uno tra sottotitolo ed abstract, che possono però anche mancarem e da una serie di para
  - <!ELEMENT sezione (titolo, sottotitolo?, abstract?, para+)>
    Come sopra, ma sottotitolo ed abstract possono coesistere

# mixed content

- Contrariamente a quanto accade in SGML, in XML il contenuto di testo #PCDATA ed il contenuto di elementi possono combinarsi solo nella forma seguente:

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

- Ogni paragrafo contiene un testo in cui si possono trovare, opzionalmente, degli elementi <bold> ed <italic>
- Es:

```
<para>
<bold>Questo</bold> testo contiene delle sezioni
in <bold>grassetto</bold> ed in <italic>corsivo</italic>,
ma potrebbe anche non averne
</para>
```

# attributi

- Il DTD permette anche di vincolare gli attributi dei singoli tag, cioè dei singoli elementi.
- In generale gli attributi vengono specificati dal costrutto ATTLIST:

```
<!ATTLIST elemento
        attributo1      tipo1     modificatore1
        attributo2      tipo2     modificatore2
        attributo-n     tipo-n    modificatore-n>
```

- I tipi definiscono l'insieme o la tipologia dei valori assumibili dall'attributo
- I modificatori identificano le condizioni di obbligatorietà o opzionalità dell'attributo ed, eventualmente, un valore di default per lo stesso

# attributi stringa

<!ATTLIST message
lang CDATA “Italiano”>

- In questo caso l’attributo lang è una stringa
- Se l’attributo è presente nel file il suo valore è quello specificato
- Altrimenti viene assunto il valore di default “Italiano”

```
<note>
…
<message lang=“English”>
Ricordati l’appuntamento
</message>
</note>
```

```
<note>
…
<message>
Ricordati l’appuntamento
</message>
</note>
```

# Attributi per enumerazione

<!ATTLIST person
salutation (Mr | Mrs | Miss | Dr) “Mr”>

- Il titolo di ogni persona assume valori nell’insieme indicato
- In mancanza del parametro si assume per default “Mr”

```
<person salutation=“Dr”>
        <name>Luke</name>
        <surname>Brown</name>
</person>
```

# Tipi di attributi predefiniti

- DTD definisce alcuni tipi speciali, che aiutano il progettista soprattutto per quanto riguarda le relazioni tra elementi
  - ID: identificativo univoco all'interno del file

```
<!ATTLIST        User
          login  ID       #REQUIRED>
```

  - IDREF: riferimento ad un identificativo univoco definito nel file

```
<!ATTLIST        User
          userClass        IDREF    #REQUIRED>
```

  - IDREFS: come IDREF, ma può esserci una lista di riferimenti
  - NMTOKEN o NMTOKENS: stringa (o lista di stringhe) di caratteri senza spazi o caratteri di interpunzione
  - ENTITY o ENTITIES: il valore deve essere un'entità

# modificatori

- Valore di default: espresso da una stringa indica il valore da assegnare all'attributo in mancanza di diverse indicazioni
- Valore fisso: definito da #FIXED più il valore. L'attributo assume obbligatoriamente il valore assegnato e l'autore del documento XML non può modificarlo:
  <!ATTLIST persona numeroGambe CDATA #FIXED "2">
- Specifica di obbligatorietà: #REQUIRED. Indica che l'attributo deve essere sempre presente in ogni elemento
  <!ATTLIST misura val CDATA #REQUIRED>
- Specifica di opzionalità: #IMPLIED. Indica che l'attributo è opzionale e può non essere specificato dall'autore del documento. Se combinato con ID indica che il sistema genererà un identificativo automaticamente

# entità

- Le entità del DTD sono frammenti ricorrenti di contenuti testuali a cui vengono associati degli identificatori che possono essere “espansi” come macro all’interno del documento prima di procedere al parsing vero e proprio
- La definizione avviene secondo lo schema:
  - <!ENTITY nomeEntità valore>
- L’utilizzo avviene inserendo nel testo la sequenza:
  - &nomeEntità;
- Esempio:
  - <!ENTITY autore “Mario Arrigoni Neri”>
  - <document>
    <title>Introduzione ad XML</title>
    <author>&autore;</author>
    </document>

# entità esterne

- Le entità esterne vengono “recuperate” da un file esterno all’XML
  <!ENTITY nomeEntità SYSTEM “url del file”>
- Es: <!ENTITY text SYSTEM “book.txt”>

```
<book>
<title>..</title>
<text>&text;</text>
</book>
```

- E’ possibile specificare entità da non parsare, tipicamente come entità di dati binari
  <!ENTITY immagine SYSTEM “immagine.gif” NDATA gif>

# entità predefinite

- Sono predefinire tutte le entità associabili ai singoli caratteri sulla base del codice (unicode) degli stessi. Ad esempio l'entità 
 corrisponde al carattere ASCII 10.
- Alcuni caratteri, tra cui '<' ed '&', creano problemi quando vengono mischiati al testo. Per questo sono predefinire alcune entità che possono sostituire questi caratteri:

| | |
|---|---|
| < | "<" |
| > | ">" |
| & | "&" |
| ' | " ' " |
| " | " " " |

- Equivalente a : <![CDATA [<]]>

# entità parametriche

- Sono entità definire nel DTD ed utilizzate all'interno del DTD stesso
- Vengono sostituite durante la lettura del DTD
- Contengono frammenti ricorrenti del modello di contenuto
- Es:
  volendo definire in un unico punto il tipo di testo che contiene sezioni in grassetto:
  - <!ENTITY % testo "#PCDATA">
  - <!ENTITY % testoConBold "(%testo;|bold)*">
  - <!ELEMENT testo %testoConBold;>