Sintassi dei DTD

- `<!DOCTYPE … >`
- `<!ELEMENT … >`
- `<!ATTLIST … >`
- `<!ENTITY … >`: Entità generali
- `<!ENTITY % … >`: Entità parametriche

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.**
- N.B.: In XML il nome del DOCTYPE deve essere il nome del tag radice.

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 content-model >`
- `Es: <!ELEMENT para (#PCDATA | bold)+ >`
- **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 XML, 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.**

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

- **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

- + *(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 il content model misto ha come UNICA FORMA la selezione ripetibile di elementi alternativi in cui il primo è #PCDATA.**

Lista di attributi: <!ATTLIST … >

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

Attributi di tipo predefinito

- `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.

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    #IMPLIED
      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  -->`
  ```

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 è un documento </doc>
  e l'uso
  <doc> Questo è 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.**

```
<!DOCTYPE mydoc [
<!ENTITY % blocco-dtd SYSTEM "blocco.dtd">
<!ENTITY grosso-testo SYSTEM "testo.xml">
%blocco-dtd;
]>
<mydoc>
  <p>&grosso-testo;</P>
</mydoc>
```

Analisi dei documenti

- Classificazione dei componenti
- Selezione dei componenti, costruzione della gerarchia, dei blocchi informativi e degli elementi di dati
- Identificazione delle connessioni
- Verifica e miglioramento iterativo delle specifiche

Tre grandi dubbi

- **Attributi o elementi?**
  - **Nei documenti di testo, la distinzione è spesso chiara: elementi per contenuto, attributi per meta-informazioni. Nell'interscambio di dati è meno chiara: la destinazione di un URL è informazione o metainformazione?**
- **Vincoli stretti o laschi?**
  - **La tentazione esiste sempre di imporre tutto quello che si può imporre. Tuttavia in XML due possono essere gli obiettivi:**
    - **Identificare semanticamente gli elementi (tutti i titoli si chiamano TITOLO)**
    - **Uniformare la struttura (tutti gli INDIRIZZO hanno una VIA, un CAP, una CITTA, una PROV). Più è restrittiva la regola, più uniforme è il risultato.**
  - **Qual è il vero obiettivo? Descrivere o regolare?**
- **Content model misti: sì o no?**
  - **I content model misti sono più difficili da trattare, descrivere, regolare. Tuttavia rappresentano una libertà di movimento che è importante lasciare agli autori.**
  - **Un elemento fatto per essere letto (ad es. DESCRIZIONE)** deve **aver content model misto: enfasi, link ipertestuali, elementi semantici individuali, ecc.**

Esercizio

Costruire un database XML di schede di alberghi. La radice del documento è un elemento *ALBERGHI*, che contiene un elemento *ALBERGO* per ogni albergo nel database (che può essere vuoto).

Ogni elemento *ALBERGO* ha un attributo identificativo *CODICE*, unico nel documento ed i seguenti nodi figlio: *NOME*, *INDIRIZZO* (padre dei nodi *VIA, CIVICO, CAP, CITTA, NAZIONE* tutti obbligatori e non ripetibili), *CATEGORIA* (con content-model vuoto ed attributo *STELLE*), *SERVIZI*, *PREZZI*.

I nodi *NOME, VIA, CIVICO, CAP, CITTA, NAZIONE* contengono testo.

Il nodo *SERVIZI* è opzionale e contiene (tutti opzionali) i nodi: *PARCHEGGIO* (elemento vuoto, con attributo *TIPO* che può assumere valori *ESTERNO* o *COPERTO* e *NUMERO* che specifica il numero di posti disponibili), *SALE_RIUNIONI* (padre di uno o più nodi *SALA*, ognuno con attributo *NOME* e *POSTI*).

Il nodo prezzi ha due figli *ALTASTAGIONE* e *BASSASTAGIONE*, che a loro volta contengo due nodi *SINGOLA* e *DOPPIA*, contenenti il relativo prezzo in formato testo.