DTD (Document Type Definition)

- NON è il linguaggio XML
- NON supporta i Namespace, ma può coesistere con loro tramite la definizione di 1 ATTRIBUTO di nome xmlns col valore dell'URI del namespace
  - ⇒ NON è possibile l'uso di prefissi

SCHEMA:

<!ELEMENT nome-elem. modello >

- dichiarazione di 1 elemento
- MODELLO definisce la struttura del contenuto:
  a) EMPTY (contenuto vuoto)
  b) ANY (contenuto qualsiasi)
  c) (#PCDATA | e_1 | e_2 | ... | e_n)* (contenuto MISTO. è l'unico modo per descriverlo. potrebbe essere anche vuoto)
  d) contenuto ELEMENTO: sequenza di elementi descritti tramite un'espressione regolare.

COMMENTI: possono comparire ovunque, tranne negli elementi a contenuto EMPTY.

NON è possibile imporre VINCOLI sui dati testuali (#PCDATA).

<!ATTLIST nome-elem. lista-definizioni >

- dichiarazione di 1 lista di attributi per l'elemento "nome-elem."
- LISTA_DEFINIZIONI è nella forma:
  nome-attr. tipo valore-default

└ TIPO di 1 attributo:

a) CDATA (stringa)
b) $(S_1|S_2|\cdots|S_N)$ con S_i di tipo stringa (tipo ENUMERAZIONE)
c) NMTOKEN (token di nome)
d) NMTOKENS (lista di token di nome separati da spazi)
e) ID (identificativo, indica 1 CHIAVE)
f) IDREF (tipo RIFERIMENTO)
g) IDREFS (lista di riferimenti a ID separati da spazi)

] CHIAVE ESTERNA / lista chiavi esterne

└ VALORI di DEFAULT di 1 attributo:

a) #REQUIRED: attributo richiesto. Se non c'è, il documento non è valido
b) #IMPLIED: attributo opzionale, nessun valore di default
c) #FIXED "valore": attributo opzionale, se presente deve essere uguale a "valore", se assente viene fissato uguale a "valore"
d) "valore": attributo OPZIONALE con valore di default.

<!ENTITY nome "definizione">

└ entità INTERNE, ovvero abbreviazioni che vengono rimpiazzate in fase di normalizzazione del DOCUMENTO ISTANZA
└ &nome; viene rimpiazzata con DEFINIZIONE

<!ENTITY % nome "definizione>

└ entità PARAMETRO INTERNE: macro applicabile solo allo SCHEMA, non al doc. istanza
└ %nome; viene rimpiazzata con DEFINIZIONE

<!ENTITY nome SYSTEM "URL">

└ entità ESTERNE: riferimento ad un'altra risorsa che può contenere o meno dati XML
└ per risorse NON XML devo indicare il FORMATO della risorsa nel seguente modo:

<!NOTATION nome_formato SYSTEM "URL-form.">

└ descrive il FORMATO dell'entità unparsed

SEZIONI CONDIZIONALI (poco utilizzate)

DOCUMENTO ISTANZA

<?xml version="1.1" encoding="..." ?> → PROLOGO OPZIONALE

<!DOCTYPE radice SYSTEM "URI">

<radice>
...
</radice>

<!DOCTYPE radice
- SYSTEM "URI">
- PUBLIC "URI" "URL">
- [definizione schema]>

XML Schema

- linguaggio XML
- offre il meccanismo di inclusione di file
- sistema di tipi gerarchico e complesso
- sistema di specifica di frammenti riutilizzabili di content model e attributi
- permette di inserire annotazioni in modo esplicito e controllato.
- estremamente VERBOSO e poco chiaro da leggere
- validazione più onerosa

XSD (XML Schema Definition)

```
<? xml version = "1.0" ? >
   < xsd : schema  xmlns: xsd = "http://www.w3.org/2011/XMLSchema"
                   targetNamespace = " URI_NS"
                   xmlns = " URI_ns"
                   elementFormDefault = "qualified" >

        definizione schema

   </xsd : schema >
```

(URI)

Convenzione: si mette il PREFISSO davanti ai tag di XML Schema

Il PREFISSO lo posso scegliere io.

xmlns: PREFISSO = "URI"
- nome del Namespace

targetNamespace = "URI" → indica il Namespace che si sta DEFINENDO

elementFormDefault = "qualified" / "unqualified" (DEFAULT)

Un elemento GLOBALE DEVE avere davanti il prefisso del namespace. Un elemento LOCALE (nello schema) lo può ereditare dal padre (UNQUALIFIED), oppure lo deve avere davanti (QUALIFIED).

→ se ho: elementFormDefault = "qualified" significa che elementi GLOBALI ed elementi LOCALI si comportano allo stesso modo.

attributeFormDefault = "qualified" / "unqualified" (DEFAULT)

<xsd:element name = "name" type = "tipo" />
- definizione di 1 ELEMENTO

TIPI
- SEMPLICI: non possono contenere Markup, né avere attributi. sequenza di caratteri. <xsd: simpleType name = "name_tipo">
- COMPLESSI: possono contenere Markup e avere attributi. <xsd:complexType name = "name_t">

Un tipo SEMPLICE può venire assegnato ad un attributo, mentre un tipo COMPLESSO no.

→ un tipo VUOTO è 1 tipo complesso.

TIPI ANONIMI: possono essere utilizzati solo per l'elemento dentro cui vengono definiti.

DERIVAZIONE DI TIPI SEMPLICI:

a) RESTRIZIONE: <xsd: restriction base = "tipo_base">
FACETS
</xsd: restriction>

b) UNIONE: <xsd: union>
TIPO 1
TIPO 2
</xsd: union>

c) LISTA: `<xsd:list itemType="nome_tipo" />`

FACETS = caratteristiche indipendenti tra di loro che impongono dei vincoli ai valori accettati per quel tipo.

DEFINIZIONE DI TIPI COMPLESSI:

a) Sequenza di elementi: `<xsd:sequence>` (A, B, C)

b) Scelta (A | B | C): `<xsd:choice>`

(possono essere ANNIDATI)

c) ALL (A & B & C): `<xsd:all>`

- tutti gli elementi devono essere presenti, ma in ordine qualsiasi
- se presente, deve essere l'UNICA struttura di 1 tipo complesso.
- validazione molto onerosa

Gli ATTRIBUTI vanno definiti DOPO gli elementi.

`<xsd:attribute name="n" type="t" fixed="v" />`

- definizione di 1 ATTRIBUTO
- va all'interno di 1 tipo complesso che corrisponde al tipo dell'elemento di cui l'attributo n sarà attributo.
- di 1 attributo posso definire:
  a) name
  b) type
  c) use ← required / optional (DEFAULT)
  d) fixed
  e) default

d) content model misti: permette di avere elementi immersi in testo semplice. È definito da:

`<xsd:complexType name="t" mixed="true">`

e) content model con ATTRIBUTI e CONTENUTO SEMPLICE:

```
<xsd:complexType name="t">
  <xsd:simpleContent>
    <xsd:extention base="tipo_semp.">
      <xsd:attribute name="n" type="t" />
    </xsd:extention>
  </xsd:simpleContent>
</xsd:complexType>
```

DERIVAZIONE DI TIPI COMPLESSI:

a) RESTRIZIONE: si limitano ulteriormente i vincoli espressi

b) ESTENSIONE: si aggiungono nuovi elementi o attributi DOPO quelli precedentemente definiti

Per DERIVARE 1 tipo complesso con 1 RESTRIZIONE lo devo RISCRIVERE completamente, con le modifiche che voglio apportare ⇒ spesso conviene riscriverlo da 0, tranne quando serve utilizzare i SOTTOTIPI

Con 1 ESTENSIONE scrivo solo gli elementi o attributi che aggiungo.

nillable="true" ⇒ l'elemento o attributo viene validato sia se ha il contenuto del tipo corretto, sia se è VUOTO