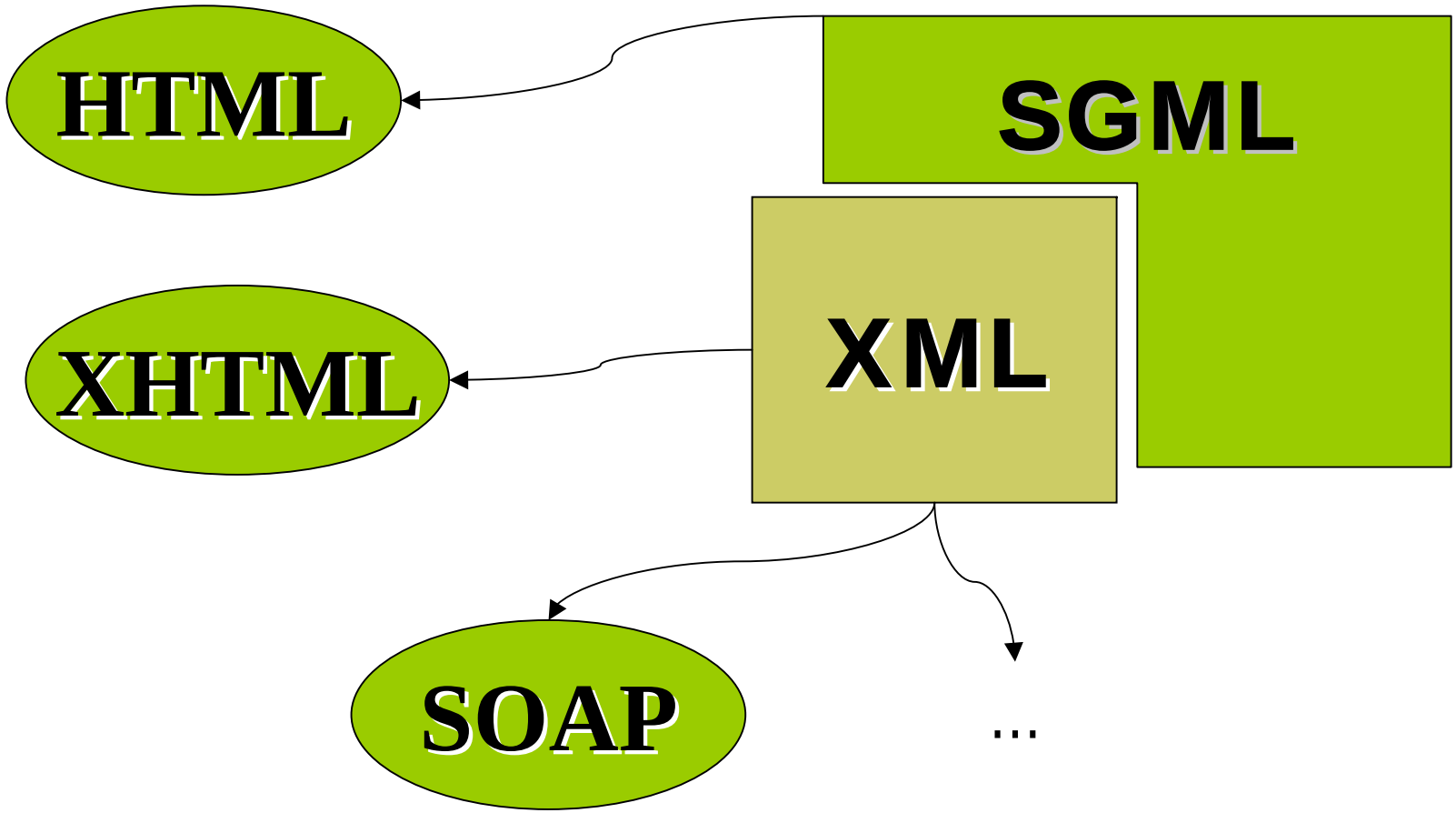
XML

Origini di XML

- XML: **linguaggio di markup estensibile**
- Deriva da **SGML**, estremamente semplificato
- Viene usato per **definire linguaggi di tipo markup**

La famiglia di XML



Un documento XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE collection SYSTEM "[URI]">
<collection xmlns="[URI]">
    <CD number="1">
        <song album="santana1" track="11">
            <title>African Bamba</title>
            <length>4:42</length>
        </song>
        <song album="santana1" track="9">
            <artist>Santana &amp;amp; Mana</artist>
            <title>Corazon Espinado</title>
            <comment>    <!CDATA[First Hit from the Album]]> </comment>
            <length>4:36</length>
        </song>
        <album ID="santana1">
                <artist>Santana</artist>
                <title>Supernatural</title>
                <year>1999</year>
        </album>
    </CD>
</collection>
```

Vantaggi

- XML permette di creare facilmente strutture ad-hoc per contenere informazione semi-strutturata
- I parser XML sono diffusi su tutte le piattaforme. E' possibile appoggiarsi a questi per decodificare e validare le strutture XML, limitandosi a gestire solo l'informazione contenuta
- XML è completamente text-based, quindi leggibile anche dagli esseri umani. Supporta UNICODE, quindi è adatto a ogni tipo di scrittura
- Essendo dati testuali strutturati come HTML, i dati XML possono essere trasportati usando il protocollo HTTP

Svantaggi

- I documenti XML, a causa delle struttura testuale e dei tag, tendono ad essere **molto più ingombranti** di quelli in formato binario
- **I parser XML non sono veloci** come quelli scritti *ad-hoc* per formati specifici, soprattutto se binari

Linguaggi e standard correlati

- DTD: specifica di vincoli semplici su documenti XML
- DOM, SAX: API per il parsing e la programmazione
- XSLT: trasformazione di documenti XML
- XLink, XPointer: collegamento tra documenti XML
- XMLSchema: specifica di vincoli complessi su documenti XML
- XPath, XQuery: interrogazione di dati XML
- RDF: specifica di meta-informazioni su risorse
- SOAP, WSDL, UDDI: strutturazione di messaggi, definizione di interfacce, resource discovery
- RSS: diffusione di notizie
- MathML: espressioni matematiche
- ebXML: electronic business
- ...

Document Type Definition (DTD)

- Sono utilizzati per **dare uno schema** ai dati XML
- E' possibile infatti definire una grammatica che vincola:
  - gli **elementi** utilizzabili
  - il modo in cui gli elementi possono essere nidificati, in che ordine e con quante occorrenze
  - gli **attributi** applicabili a ciascun elemento, la loro obbligatorietà ed i valori predefiniti

Parsing - DOM

- Con le API Document Object Model (DOM) il parsing produce una rappresentazione ad albero in memoria
  - Gerarchia di oggetti “Node”: documento, elemento, attributo, testo, ecc.
- E’ possibile manipolare i dati
- Consuma memoria

Parsing - SAX

- Con le API Simple API for XML (SAX) il parsing produce un flusso di eventi
  - Eventi “inizio tag”, “fine tag”, “testo”, ecc.
- Non è necessario caricare un intero documento in memoria
  - Permette di ignorare i dati non di interesse
- Più veloce, non consuma memoria
- Non è possbile manipolare i dati se non costruendo un albero in memoria sulla base degli eventi ricevuti

Dichiarazione XML

- La dichiarazione XML è **obbligatoria** e deve essere posta all'inizio del documento:

```
<?xml version="1.0" encoding="UTF-8"
standalone="no" ?>
```

- Gli attributi sono:
  - ***version*** (*obbligatorio*): la **versione di XML** usata
  - ***encoding*** (*opzionale*): nome della **codifica dei caratteri** usata nel documento
  - ***standalone*** (*opzionale*): se vale *yes* indica che il file non fa **riferimento a file esterni** (default: no)

Dichiarazione DOCTYPE

- La dichiarazione è obbligatoria se il documento deve essere validato

  <!DOCTYPE *RootElement ExternalDTDReference* [*InternalDTDSubset*]>

- ***RootElement*** (*obbligatorio*) è il nome dell'elemento radice del documento
- ***ExternalDTDReference*** (*opzionale*) è SYSTEM "*uri*", dove *uri* punta a un file esterno contenente la definizione del linguaggio usato nel documento
- ***InternalDTDSubset*** (*opzionale*) è un DTD specificato nel documento

Processing Instructions

- Le Processing Instructions (PI) vengono usate per **passare informazioni extra-markup ai programmi che manipoleranno il file XML**
- Possono apparire ovunque, dopo la dichiarazione XML

<?target data ?>

- ***target*** identifica il **destinatario** della PI
- ***data*** è una **stringa di dati** per la PI

Commenti

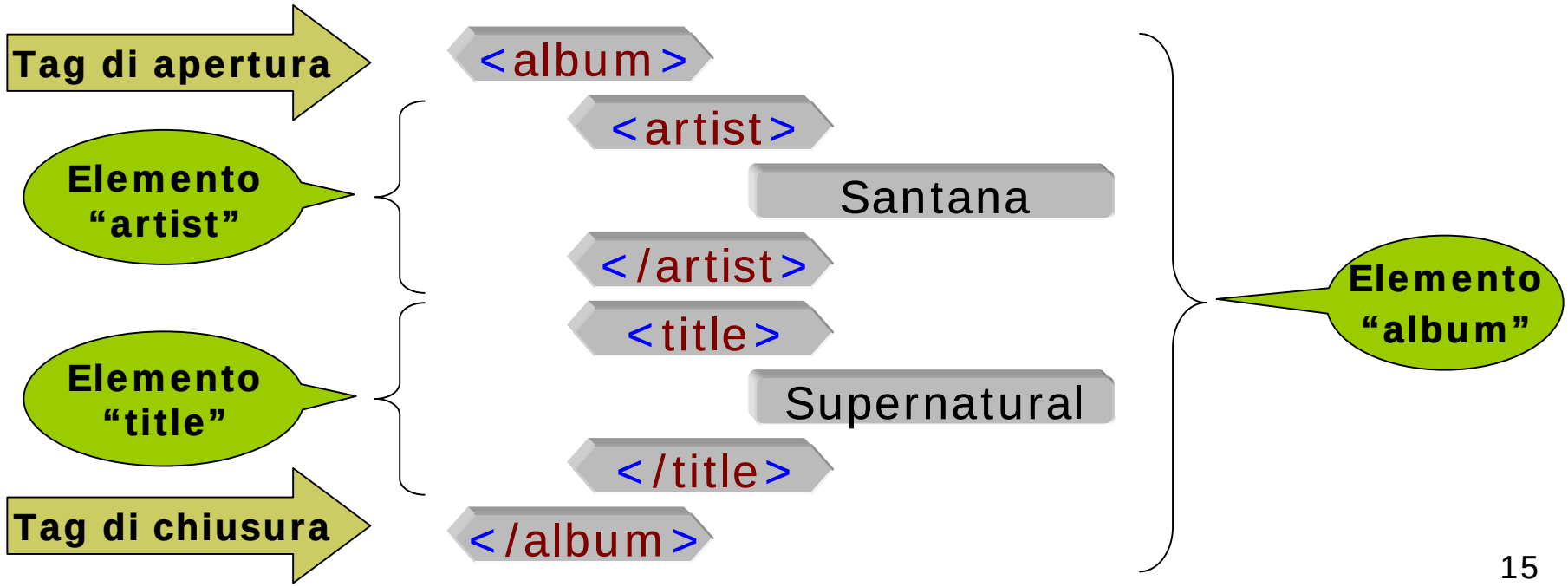
- Possono apparire ovunque tranne che all'interno degli attributi

```
<!-- this is a comment -->
```

- Il commento è chiuso da un `-->`,che quindi non può apparire nella stringa interna

Elementi

- Gli **elementi** sono alla base della strutturazione dei documenti XML
- Un elemento è un **frammento di dati**, ***limitato*** ed ***indentificato*** (tramite un nome) da *tag*



Elementi: regole

- I nomi degli elementi sono **case-sensitive**
- **Ogni elemento aperto deve essere chiuso** entro la fine del documento
- Gli elementi possono essere nidificati, e in tal caso **vanno chiusi esattamente nell'ordine inverso a quello di apertura**
- Un documento XML deve avere **un unico elemento radice**, in cui tutti gli altri sono nidificati

Elementi: sintassi

- Il **tag di apertura** di un elemento ha la forma seguente:

  <*nome attributi*>

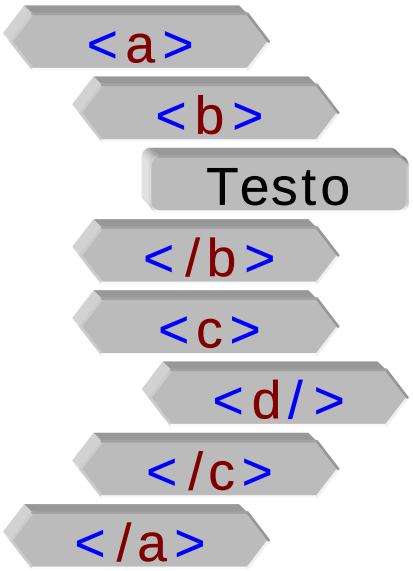
  - ***nome*** è il nome dell'elemento
  - ***attributi*** è una lista **opzionale** di attributi per l'elemento
- Il **tag di chiusura** corrispondente ha la forma seguente:

  </*nome*>
- Alcuni elementi possono essere **privi di contenuto**; in questo caso è possibile omettere il tag di chiusura scrivendo quello di apertura come segue (*empty tag*):

  <*nome attributi* />

Elementi: gerarchia

- Gli elementi, nidificandosi, creano una struttura ad albero
- All'interno di questa struttura si definiscono alcuni "rapporti di parentela" utili per individuare gli elementi:



- *a* è il nodo **radice**
- *b* e *c* sono figli di *a*, il *Testo* è **figlio** di *b*, *d* è figlio di *c*
- *c* è il **padre** di *d*, *b* è il padre del *testo*, a è il padre di *b* e *c*
- *b* e *c* sono **fratelli**
- *b*, *c*, *d* e il *testo* sono discendenti di *a*, *d* è un **discendente** di *c*, il *testo* è un discendente di *b*
- *a* è un **predecessore** di *b*, *c*, *d* e del *testo*, *b* è un predecessore del *testo*, *c* è un predecessore di *d*

Esempio

```
<message>
   <to>you@yourAddress.com</to>
   <from>me@myAddress.com</from>
   <subject>XML Is Really Cool</subject>
   <text>    How many ways is XML cool? Let me count the
    ways...  </text>
</message>
```

Sezioni CDATA

- Permettono di definire esplicitamente **aree in cui si trova solo testo semplice**

<!CDATA[<<solo testo!>>]]>

- Sono utili per impedire che il parser consideri markup alcune stringhe che ne hanno solo la forma

Attributi

- Gli attributi permettono di specificare **proprietà degli elementi**, come coppie nome-valore
- Sono usati per definire proprietà che non possono o non si vogliono inserire nel contenuto dell'elemento
- Vengono specificati all'interno dei tag di apertura degli elementi
- L'ordine non è significativo

Attributi: regole

- I nomi degli attributi sono **case-sensitive**
- Lo stesso tag non può contenere due attributi con lo stesso nome
- Non sono ammessi attributi senza valore
- Il valore degli attributi deve essere specificato **tra virgolette semplici o doppie**
- Il valore non può contenere markup, sezioni CDATA o virgolette uguali a quelle iniziali

Attributi: sintassi

- **Sintassi di base** usata all'interno dei tag di apertura:

  <nome attributo="valore">

- Una **lista di attributi** si ottiene elencando più attributi separati da uno o più spazi:

  <nome att1="vl1" att2="vl2">

- Per includere **virgolette** nel valore, è necessario usare un tipo diverso da quello usato per delimitare il valore stesso:

  <nome att1=' "virgolette" '>

Esempi (1)

```
<message to="you@yourAddress.com"
         from="me@myAddress.com"
         subject="XML Is Really Cool">
   <flag/>
   <text> How many ways is XML cool? Let me count the
            ways...  </text>
</message>
```

Esempi (2)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE collection SYSTEM "[URI]">
<collection xmlns="[URI]">
    <CD number="1">
        <song album="santana1" track="11">
            <title>African Bamba</title>
            <length>4:42</length>
        </song>
        <song album="santana1" track="9">
            <artist>Santana &amp;amp; Mana</artist>
            <title>Corazon Espinado</title>
            <comment>    <!CDATA[First Hit from the Album]]> </comment>
            <length>4:36</length>
        </song>
        <album ID="santana1">
                <artist>Santana</artist>
                <title>Supernatural</title>
                <year>1999</year>
        </album>
    </CD>
</collection>
```

Namespaces

- I namespaces servono a dichiarare **l'appartenenza degli elementi a particolari linguaggi XML**
- Sono particolarmente utili se **più linguaggi vengono mescolati nello stesso documento**, con possibili di collisioni tra nomi
- **Ogni elemento può contenere dichiarazioni di namespaces**, la cui validità è estesa a tutto il contenuto dell'elemento stesso
- La **dichiarazione del namespace** viene inserita nei tag di apertura, in modo simile a un attributo

Namespaces: sintassi

- Questa dichiarazione di namespace indica che tutti gli elementi il cui nome è prefissato da “prfx:” andranno considerati appartenenti al namespace puntato da *uri*:

  <name xmlns:*prfx*="*uri*">

- La **dichiarazione di namespace standard** indica il namespace per tutti gli elementi non prefissati:

  <name xmlns="*uri*">

- Ci possono essere **più namespaces** attivi per lo stesso elemento, ma solo uno standard:

  <name xmlns="*uri1*" xmlns:*prfx1*="*uri2*" xmlns:*prfx2*="*uri3*">

Entità (1)

- I documenti XML sono costituiti da una serie di **entità**. Il documento stesso è una entità
- Tutte le entità, tranne il documento e il DTD esterno, hanno un nome
- Le **entità parsed** sono quelle più comuni, e il parser XML le sostituisce sempre col loro testo di definizione
- Le **entità unparsed** possono contenere **qualsiasi tipo di dato**, anche non testuale. Il parser XML non le analizza, e sono accessibili solo usando le ***notazioni***

Entità (2)

- Le **entità parsed** consentono di **inserire stringhe nel documento facendo riferimento a una definizione separata**, invece di scriverle esplicitamente
- Sono utili ad esempio
  - nei casi in cui ci siano **caratteri non digitabili** direttamente
  - per **espandere stringhe usate di frequente**
  - per scrivere **caratteri che non sono ammessi in maniera esplicita in un contesto**, perché riservati (come le virgolette o i segni ‘<’ e ‘>’)

Entità parsed: sintassi

- Le **entità generali**, che possono rappresentare stringhe qualsiasi, sono definite nel DTD e si richiamano nel documento XML con la sintassi:

  &nome;

  - dove ***nome*** è il nome dell'entità
- Le **entità carattere**, che rappresentano singoli caratteri, si richiamano con la sintassi:

  &#numero;

  - dove ***numero*** è il codice decimale UNICODE per il carattere

  &#xnumero;

  - dove ***numero*** è il codice esadecimale UNICODE per il carattere

Testo

- Il testo inseribile nei documenti XML **comprende tutti i caratteri definiti in UNICODE**.
- Non è possibile usare esplicitamente i caratteri ‘>’, ‘<’ e ‘&’, per i quali è sempre necessario usare le corrispondenti entità carattere

Validazione di documenti XML

- **Un documento XML è *ben formato*** se rispetta le regole generali di sintassi
- **Un documento XML è *valido*** se è ben formato e rispetta le regole sintattiche contenute nel DTD associato
- Esistono **parser validanti** e **non validanti**
  - Questi ultimi possono ignorare l'eventuale DTD, tranne le dichiarazioni di entità generali

Un DTD

```
<!ELEMENT artist (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>

<!ELEMENT album (artist, title, year?)>
<!ATTLIST album
   ID ID #REQUIRED
   media CDATA #IMPLIED "CompactDisc“
>
```

Dichiarazione di entità generali (1)

<!ENTITY *name content*>

- ***name*** è il nome dell'entità da definire
- ***content*** definisce il contenuto dell'entità e può essere:
  - "*valore*"

    Il valore dell'entità è la stringa specificata, l'entità è di tipo **parsed** e si dice **interna**

Dichiarazione di entità generali (2)

- SYSTEM “*uri*”
  L’entità è **parsed** e **esterna**, cioè il suo valore deve essere **prelevato da un file** separato, puntato da *uri*
- PUBLIC “*pubid*” “*uri*”
  L’entità è **parsed** e **esterna**, e *pubid* è un identificatore alternativo per il file puntato da *uri* che contiene il valore dell’entità

Dichiarazione di entità parametriche

- Le **entità parametriche**, sono simili a quelle generali, ma **possono essere usate solo all'interno del DTD** (rendendolo "parametrico")

<!ENTITY % *name content*>

- ***name*** è il nome dell'entità da definire. All'interno del DTD, **le entità parametriche vengono richiamate con la sintassi**

%*name*;

- ***content*** definisce il contenuto dell'entità

Dichiarazione di elementi

<!ELEMENT *name content-model*>

- ***name*** è il nome dell'elemento da definire
- ***content-model*** definisce gli elementi nidificabili in quello definito, e può essere:
  - EMPTY: l'elemento non può contenere **nulla**
  - ANY: l'elemento può contenere **testo e ogni altro elemento**
  - un **modello di contenuto**, se l'elemento può contenere solo altri elementi
  - un **modello misto**, se l'elemento può contenere anche testo

Elementi: modello di contenuto

<!ELEMENT *persona (titolo?, nome,cognome, (indirizzo | telefono)*)*>

- Il modello di contenuto è simile ad una **espressione regolare**
- **Ogni nome di elemento è anche un modello valido**. Esso indica che l'elemento definito deve contenere **esattamente un elemento del tipo dato**
- Se *p* e *q* sono due modelli validi allora lo sono anche:
  - (*p*) **raggruppamento** (equivale a *p*)
  - *p* | *q* **disgiunzione** (p oppure q)
  - *p*, *q* **concatenazione** (p e poi q)
  - *p** **star** (zero o più volte p)
  - *p*+ **croce** (una o più volte p)
  - *p*? **opzione** (p oppure nulla)

Elementi: modelli misti

<!ELEMENT *testo (#PCDATA | nota)*>

- Il **modello misto** si usa per gli elementi che **devono contenere anche testo semplice**
- **Il contenuto testuale si indica con #PCDATA**
- Se è si vogliono nidificare altri elementi, non è possibile specificarne l'ordine o le ripetizioni. In altre parole, **si può usare solo l'operatore di disgiunzione**

Esempio

```
<!ELEMENT slideshow (slide+)>
<!ELEMENT slide (title, item*)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT item (#PCDATA | item)* >
```

Dichiarazione di attributi

```
<!ATTLIST name
   att-name att-type att-default
   ...>
```

- ***name*** è il nome dell'elemento di cui si definiscono gli attributi
  - Tutte le dichiarazioni ATTLIST per lo stesso elemento vengono fuse
  - In caso di conflitti su attributi ridefiniti, la prima definizione ha la precedenza
- ***att-name*** è il nome di un attributo
- ***att-type*** è il tipo dell'attributo
- ***att-default*** indica se l'attributo è obbligatorio, qual è il suo default, ecc.
- Si possono dichiarare un numero arbitrario di attributi in ogni ATTLIST

Attributi: tipi (1)

- I valori ammissibili per il campo ***att-type*** sono:
  - CDATA **testo generico**
  - ID **id univoci**
    - Un elemento può avere al più un attributo di tipo ID
    - I valori degli attributi ID di tutti gli elementi in un documento XML devono essere univoci
  - IDREF **riferimento a valore ID**
    - Un attributo di questo tipo può avere come valore solo uno dei valori degli attributi di tipo ID dello stesso documento

Attributi: tipi (2)

- IDREFS **riferimenti a valori id**
  - IDREF separati da spazi
- ENTITY **nome di una entità**
  - I valori devono essere quelli di **entità unparsed** dichiarate nel DTD
- ENTITIES **nomi di più entità**
- NMTOKEN **parola 'nmtoken'**
  - Può contenere solo lettere, numeri e i simboli '-', '_', '.' e ':'.
- NMTOKENS **sequenza di nmtoken**

Attributi: tipi (3)

- (nmtoken | ...) **lista valori**
  - Il valore dell'attributo deve essere uno di quelli della lista in *or*
- NOTATION (*nt-ref*) **entità *unparsed***
  - *nt-ref* è una **notazione** che specifica il tipo dell'**entità esterna unparsed**

Attributi: default

- ***att-default*** deve avere una delle forme seguenti:
  - #REQUIRED
    - L'attributo è **obbligatorio**, e deve essere sempre specificato
  - #IMPLIED
    - L'attributo è **facoltativo**
  - "*valore*"
    - L'attributo è **facoltativo**
    - Se non viene specificato è come se fosse stato specificato col *valore* dato
  - #FIXED "*valore*"
    - L'attributo **può assumere *solo* il valore specificato**
    - Se non viene specificato, si assume che abbia il valore specificato

Esempio

```
<!ELEMENT slideshow (slide+)>
<!ATTLIST slideshow
    title    CDATA    #REQUIRED
    date     CDATA    #IMPLIED
    author   CDATA "unknown">
```

Dichiarazione di notazioni

<!NOTATION *name content*>

- Le notazioni permettono di **identificare con un nome il formato delle entità unparsed** o l'applicazione che può gestirle
- ***name*** è il nome della notazione
- ***content*** identifica in qualche modo il tipo delle entità associate. Può assumere uno dei seguenti valori:
  - SYSTEM "*uri*"
  - PUBLIC "*pubid*"
  - PUBLIC "*pubid*" "*uri*"

Estensione di un DTD

- **Un documento può avere sia un DTD esterno sia un subset interno**
- In questo caso si dice che il frammento di DTD interno ***estende*** quello esterno
- Nel subset interno si possono:
  - **definire nuovi elementi**
  - **definire nuove ATTLIST**, che verranno eventualmente fuse con quelle preesistenti
  - **ridefinire le entità parametriche** del DTD esterno

Esempio (1)

```
<!ELEMENT artist (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT length (#PCDATA)>

<!ELEMENT album (artist, title, year?)>
<!ATTLIST album
   ID ID #REQUIRED
   media CDATA #IMPLIED "Compact disc"
>
```

Esempio (2)

```
<!ELEMENT song (artist?, title, year?, comment?, length)>
<!ATTLIST song
   genre (Rock|Pop|Classical|Jazz|Dance|Tecno) #IMPLIED
   ID ID #IMPLIED
   album IDREF #IMPLIED
   track CDATA #IMPLIED
>
<!ELEMENT CD (album | song)+>
<!ATTLIST CD
   number CDATA #REQUIRED
   owner CDATA #FIXED "[nome del proprietario]"
>
<!ELEMENT collection (CD+)>
```