XML

XML

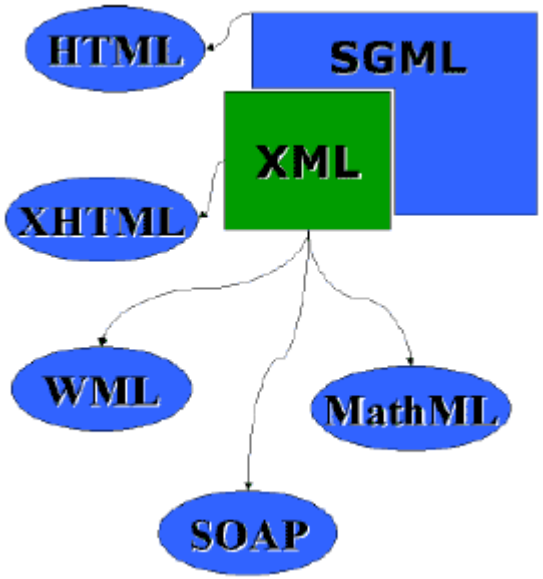
- XML (***eXtensible Markup Language)*** è un meta linguaggio.

- Può essere definito come un insieme di regole e convenzioni che consentono di descrivere qualunque linguaggio di markup.

- Esso è quindi basato su **marcatori** che possono essere definiti in base alle proprie esigenze.

Origini

- L'idea è in parte derivata dai concetti di base di SGML (Standard Generalized Markup Language).



- XML è un linguaggio a marcatori estendibile.
- I linguaggi a marcatori consentono di descrivere con precisione qualsiasi tipo di informazione: **gerarchica, lineare, relazionale o binaria**.
- In XML l'informazione viene organizzata utilizzando una **struttura gerarchica** che è possibile scorrere e navigare con semplicità e all'interno della quale è possibile ricercare le informazioni desiderate.

- XML non definisce la propria collezione di marcatori (a differenza di HTML) ma definisce le regole sintattiche attraverso le quali è possibile generare dei marcatori personalizzati e i loro eventuali attributi.

- La sintassi di XML è formata essenzialmente da tag i quali possono avere attributi ed eventualmente al proprio interno altri tag.
- I tag devono essere a coppie, ci deve essere la presenza contemporanea dei tag di apertura e chiusura.
- La presenza dei tag è necessaria (così come in HTML) per dividere il contenuto informativo del documento dalla sintassi utilizzata per rappresentarlo.

- La struttura di un documento XML è gerarchica e ad albero.
- Es.

```
<?xml version="1.0"?>
<tag1>
  <tag2> contenuto informativo </tag2>
</tag1>
```

- HTML ed XML hanno una relazione molto stretta. Infatti è possibile scrivere un documento HTML in XML.

```
<?xml version="1.0"?>
<html>
  <head>
   <title> Titolo del documento HTML</title>
  </head>
  <body>
   Un documento HTML scritto in XML
  </body>
</html>
```

Diamo al documento appena creato estensione xml

Regole di base

```
<?xml version="1.0"?>
<Gran_Premio>
    <Nazione>Belgio</Nazione>
    <Circuito>SPA</Circuito>
    <Anno>2001</Anno>
    <Griglia_Di_Partenza>
          <Pilota>
                    <Posizione>1</Posizione>
                    <Nome>Juan Pablo Montoya</Nome>
                    <Vettura>Williams BMW</Vettura>
                    <Tempo>1.52.072<Tempo>
          </Pilota>
          <Pilota>
                    <Posizione>2</Posizione>
                    <Nome>Ralph Shumacher</Nome>
                    <Vettura>Williams BMW</Vettura>
                    <Tempo>1.53.279<Tempo>
          </Pilota>
          <Pilota>
                    ...
          </Pilota>

    </Griglia_Di_Partenza>
</Gran_Premio>
```

Identifica il tipo di documento e
Specifica la versione di XML utilizzata

Tag di apertura e chiusura

Regole di base

```
<?xml version="1.0"?>
<Gran_Premio>
    <Nazione>Belgio</Nazione>
    <Circuito>SPA</Circuito>
    <Anno>2001</Anno>
    <Griglia_Di_Partenza>
          <Pilota>
                    <Posizione>1</Posizione>
                    <Nome>Juan Pablo Montoya</Nome>
                    <Vettura>Williams BMW</Vettura>
                    <Tempo>1.52.072<Tempo>
          </Pilota>
          <Pilota>
                    <Posizione>2</Posizione>
                    <Nome>Ralph Shumacher</Nome>
                    <Vettura>Williams BMW</Vettura>
                    <Tempo>1.53.279<Tempo>
          </Pilota>
          <Pilota>
                    ...
          </Pilota>

    </Griglia_Di_Partenza>
</Gran_Premio>
```

Identifica il tipo di documento e
Specifica la versione di XML utilizzata

Tag di apertura e chiusura

Elementi

Regole di base

```
<?xml version="1.0"?>
<Gran_Premio>
    <Nazione>Belgio</Nazione>
    <Circuito>SPA</Circuito>
    <Anno>2001</Anno>
    <Griglia_Di_Partenza>
          <Pilota>
                    <Posizione>1</Posizione>
                    <Nome>Juan Pablo Montoya</Nome>
                    <Vettura>Williams BMW</Vettura>
                    <Tempo>1.52.072</Tempo>
          </Pilota>
          <Pilota>
                    <Posizione>2</Posizione>
                    <Nome>Ralph Shumacher</Nome>
                    <Vettura>Williams BMW</Vettura>
                    <Tempo>1.53.279</Tempo>
          </Pilota>
          <Pilota>
                    ...
          </Pilota>

</Griglia_Di_Partenza>
</Gran_Premio>
```

Identifica il tipo di documento e
Specifica la versione di XML utilizzata

Tag di apertura e chiusura

Elementi

Il tag radice contiene tutti gli altri.
Questo viene chiamato "root element"

```
<?xml
  version="1.0"
  encoding="UTF-8"
  standalone="yes"?>
<!-- questo e' un commento, questo documento XML descrive la
     gerarchia del corso Web II e la descrizione dei temi
     trattati -->
<Corso
  docente="Alfredo Pulvirenti"
  nome_corso="Web 2"
  locazione="laboratorio PC Wyeth CT">
  <Introduzione>...</Introduzione>
  <Temi>
    <Tema
       numero="1"
       titolo="Uno sguardo al Web">
       ...
    </Tema>
    <Tema
       numero="2"
       titolo="Il linguaggio HTML">
       ...
    </Tema>
    <Tema
       numero="3"
       titolo="Linguaggi per il web server side: il PHP">
       ...
    </Tema>
    <Tema
       numero="4"
       titolo="XML">
       ...
    </Tema>
    <Tema
       numero="5"
       titolo="XHTML">
       ...
    </Tema>
    <Tema
       numero="6"
       titolo="Linguaggi per il web client side: Javascript">
       ...
    </Tema>
</Temi>
</Corso>
```

Componenti di un doc XML

- Prologo
  - È costituito da tutta la parte del documento XML che precede l'elemento root.
  - Gli attributi sono:
  - **version**: (obbligatorio) la versione di XML usata.
  - **encoding**: (opzionale) nome della codifica dei caratteri usata nel documento. (default: UTF-8 o 16)
  - **standalone**: (opzionale) se vale yes indica che il file non fa riferimento ad altri file esterni. (default: no)

Componenti di un doc XML

```
<?xml
  version="1.0"
  encoding="UTF-8"
  standalone="yes"?>
<!-- questo e' un commento, questo documento XML descrive la
     gerarchia del corso Web II e la descrizione dei temi
     trattati -->
<Corso
  docente="Alfredo Pulvirenti"
  nome_corso="Web 2"
  locazione="laboratorio PC Wyeth CT">
  <Introduzione>...</Introduzione>
  <Temi>
    <Tema
       numero="1"
       titolo="Uno sguardo al Web">
       ...
    </Tema>
    <Tema
       numero="2"
       titolo="Il linguaggio HTML">
       ...
    </Tema>
    <Tema
       numero="3"
       titolo="Linguaggi per il web server side: il PHP">
       ...
    </Tema>
    <Tema
       numero="4"
       titolo="XML">
       ...
    </Tema>
    <Tema
       numero="5"
       titolo="XHTML">
       ...
    </Tema>
    <Tema
       numero="6"
       titolo="Linguaggi per il web client side: Javascript">
       ...
    </Tema>
</Temi>
</Corso>
```

- Elemento radice

Componenti di un doc XML

```
<?xml-stylesheet
  type="text/css"
  href="esempio.css"?>
```

- Questa istruzione può essere presente nle prologo dopo la dichiarazione XML, associa un foglio di stile al documento xml.

```
<!DOCTYPE Corso SYSTEM
"corso.dtd">
```

- Ad un documento XML possono essere associate regole grammaticali. Queste ne descrivono gli aspetti semantici e consentono l'eventuale validazione automatica.

Regole fondamentali

- 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

Sintassi

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

```
<nome attributi>
```

  - *nome* è il nome dell'elemento.
  - *attributi* è una lista di attributi per l'elemento (che può non apparire).
  - Il **tag di chiusura** corrispondente ha la forma seguente:

```
</nome>
```

Notazione abbreviata dei tag senza valori

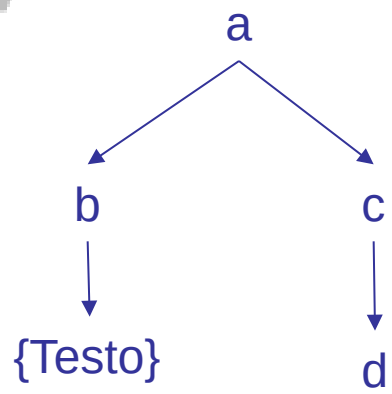
```
<nome/>
```

Gerarchia

Gli elementi, nidificandosi, creano la struttura ad albero tipica dei documenti XML. All'interno di questa struttura si definiscono alcuni "rapporti di parentela" utili per individuare gli elementi:

```
<a>
  <b>
    Testo
  </b>
  <c>
    <d/>
  </c>
</a>
```

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*.



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.

Al contrario degli elementi, per gli attributi l'ordine di presentazione non è significativo.

```
<?xml version="1.0"?>
<Gran_Premio>
    <Nazione>Belgio</Nazione>
    <Circuito>SPA</Circuito>
    <Anno>2001</Anno>
    <Griglia_Di_Partenza>
          <Pilota Posizione="1" Tempo="1.52.072">
                <Nome>Juan Pablo Montoya</Nome>
                <Vettura>Williams BMW</Vettura>
          </Pilota>
          <Pilota Posizione="2" Tempo="1.53.279">
                <Nome>Ralph Shumacher</Nome>
                <Vettura>Williams BMW</Vettura>
          </Pilota>
</Griglia_Di_Partenza>
</Gran_Premio>
```

Regole Generali

- 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 (solo nome).
- Il valore degli attributi deve essere specificato **tra virgolette semplici o doppie**.
- Il valore può contenere **riferimenti ad entità**.
- Il valore non può contenere markup, sezioni CDATA o virgolette uguali a quelle iniziali.

Sintassi attributi

```
<nome-elemento attributo="valore">
```

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

```
<nome-elemento att1="vl1" att2="vl2">
```

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

```
<nome-elemento att1=' "virgolette" '>
```

- Si possono includere **riferimenti a entità** nel valore:

```
<nome-elemento att1="&quot; salve &quot;">
```

- Alcune entità sono predefinite nel linguaggio XML e permettono di inserire quei caratteri che altrimenti sarebbero inutilizzabili. Le entità predefinite sono le seguenti:

entità	Significato
&amp;	&
&lt;	<
&gt;	>
&apos;	‘
&quot;	“

```
<?xml
  version="1.0"
  encoding="UTF-8"
  standalone="yes"?>
<Corso>
  <docente>Alfredo Pulvirenti</docente>
  <titolo>Web 2</titolo>
  <locazione>laboratorio PC Wyeth CT</locazione>
  <Descrizione>...</Descrizione>
  <Temi>
    <Tema>
       <numero>1</numero>
       <titolo>Uno sguardo al Web</titolo>
       <testo> ... </testo>
    </Tema>
    <Tema>
       <numero>2</numero>
       <titolo> Il linguaggio HTML</titolo>
      <testo> ... </testo>
    </Tema>
    <Tema
       <numero>3</numero>
       <titolo>Linguaggi per il web server side: il
     PHP</titilo>
       <testo> ... </testo>
    </Tema>
    <Tema
       <numero>4</numero>
       <titolo>XML</titolo>
       <testo> ... </testo>
    </Tema>
    <Tema
       <numero>5</numero>
       <titolo>XHTML</titolo>
       <testo> ... </testo>
    </Tema>
    <Tema
       <numero>6</numero>
       <titolo>Linguaggi per il web client side:
     Javascript</titolo>
       <testo> ... </testo>
    </Tema>
</Temi>
</Corso>
```

Grammatiche DTD

DTD (Document Type Definition): estendere XML con nuove regole grammaticali:

```
<!DOCTYPE Corso [
<!ELEMENT Corso (docente,titolo,
    locazione,Descrizione,Temi)>
<!ELEMENT docente (#PCDATA)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT Descrizione (#PCDATA)>
<!ELEMENT Temi (Tema+)>
<!ELEMENT Tema (numero,titolo,testo)>
<!ELEMENT numero (#PCDATA)>
<!ELEMENT testo  (#PCDATA)>
]>
```

Grammatiche DTD

```
<?xml version="1.0">
<!DOCTYPE Corso [
<!ELEMENT CORSO (docente,titolo,
    locazione,Descrizione,Temi)>
<!ELEMENT autore (#PCDATA)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT Descrizione (#PCDATA)>
<!ELEMENT Temi (Tema+)>
<!ELEMENT Tema (numero,titolo,testo)>
<!ELEMENT numero (#PCDATA)>
<!ELEMENT testo  (#PCDATA)>
]>
...
```

```
<?xml version="1.0">
<!DOCTYPE Corso SYSTEM "grammatica.DTD">
...
```

- Una grammatica DTD è costituita da un insieme di regole che stabiliscono la composizione e la struttura di un documento XML.
- La grammatica può essere inserita direttamente nel documento XML, oppure in in file separato e richiamato dal documento xml

Elementi della grammatica

- Sintassi

  `<!ELEMENT nome_elemento (tipologia)>`

- Informazioni testuali

```
<Indirizzo>
  Via Etnea, 10
</Indirizzo>


<!ELEMENT Indirizzo (#PCDATA) >
```

- Ulteriori elementi (elementi annidati)

  `<!ELEMENT elemento_padre(f_1,f_2,f_3)>`

```
...
<dati_anagrafici>
  <nome>Mario</nome>
  <cognome>Rossi</cognome>
  <indirizzo>Via Etnea,10</indirizzo>
<dati_anagrafici>
```

```
<!ELEMENT dati_anagrafici
    (nome,cognome,indirizzo)>

<!ELEMENT nome (#PCDATA)>
<!ELEMENT cogome (#PCDATA)>
<!ELEMENT Indirizzo (#PCDATA) >
```

Elementi della grammatica

- Informazioni testuali ed elementi annidati
- Elementi vuoti

Esempio completo

```
<?xml version="1.0"?>
<!DOCTYPE dati_anagrafici [
<!ELEMENT dati_anagrafici (nome,cognome,indirizzo)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT indirizzo (#PCDATA) >
]>
<dati_anagrafici>
  <nome>Mario</nome>
  <cognome>Rossi</cognome>
  <indirizzo>Via Etnea,10</indirizzo>
</dati_anagrafici>
```

Esempio completo

```
<?xml version="1.0"?>
<!DOCTYPE dati_anagrafici [
<!ELEMENT dati_anagrafici (nome,cognome,indirizzo)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT indirizzo (#PCDATA) >
]>
<dati_anagrafici>
  <nome>Mario</nome>
  <cognome>Rossi</cognome>
  <indirizzo>Via Etnea,10</indirizzo>
</dati_anagrafici>
```

Oltre alla verifica della correttezza spesso è necessario riconoscere se un documento XML è conforme ad una struttura predefinita.

http://validator.w3.org/#validate_by_input

Numero degli elementi di una grammatica

```
<!DOCTYPE dati_anagrafici [
<!ELEMENT dati_anagrafici
    (nome+,cognome?,indirizzo*)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT Indirizzo (#PCDATA) >
]>
```

```
<!DOCTYPE dati_anagrafici [
<!Element dati_anagrafici
    (nome+,cognome?,indirizzo*,
    (codice_fiscale | partita_iva))>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT Indirizzo (#PCDATA)>
<!ELEMENT codice_fiscale (#PCDATA)>
<!ELEMENT partita_iva (#PCDATA)>
]>
```

```
<!DOCTYPE dati_anagrafici [
<!Element dati_anagrafici
    (nome+,cognome?,indirizzo*,
   (codice_fiscale | partita_iva)*)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT Indirizzo (#PCDATA)>
<!ELEMENT codice_fiscale (#PCDATA)>
<!ELEMENT partita_iva (#PCDATA)>
]>
```

- Il simbolo + affianco all'elemento nome, indica che il campo sarà presente una o più volte (obbligatorio una volta).
- Il simbolo * affianco a indirizzo indica che il campo sarà presente zero o più volte (facoltativo).
- Il simbolo ? affianco al cognome indica che esso è facoltativo solo una volta.

- Il simbolo | indica che l'elemento potrà essere o codice_fiscale o partita_iva.
- L'ultimo esempio indica che gli elementi partita_iva e codice_fiscale potranno essere presenti contemporaneamente.

Sintassi generale

```
<!ELEMENT name content-model>
```

- ***name*** è il nome dell'elemento da definire. Un elemento già definito non può essere ridefinito.
- ***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.

- **Modello di Contenuto**

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

- Il modello di contenuto è simile a 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)

- **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.**

```
<?xml version="1.0"?>
<!DOCTYPE Autore [
<!ELEMENT Autore (Nome,Cognome,pubblicazione+)>
<!ELEMENT Nome (#PCDATA)>
<!ELEMENT Cognome (#PCDATA)>
<!ELEMENT pubblicazione (titolo, abstract, tipoPub+,coautori*)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT abstract (#PCDATA)>
<!ELEMENT tipoPub (rivista|conferenza|libro)>
<!ELEMENT coautori (#PCDATA)>
<!ELEMENT rivista (Nome,impactf,volume)>
<!ELEMENT impactf (#PCDATA)>
<!ELEMENT volume (#PCDATA)>
<!ELEMENT conferenza(titolo,luogo,data)>
<!ELEMENT luogo (#PCDATA)>
<!ELEMENT data (#PCDATA)>
<!ELEMENT libro (editore,data)>
<!ELEMENT editore (#PCDATA)>
<!ENTITY abs SYSTEM "absract.txt">
]>
<autore>
  <nome>Filippo</nome>
  <cognome>Gialli</cognome>
  <pubblicazione>
    <titolo>WEB III</titolo>
    <abstract>&abs;</abstract>
    <tipoPub>
      <libro>
        <editore>APOGEO</editore>
        <data>10/10/2009</data>
      </libro>
    </tipoPub>
    <coautori>Enzo Gialli</coautori>
  </pubblicazione>
</autore>
```

- Autore
  - Nome TESTO
  - Cognome TESTO
  - Pubblicazione ELEMENTO (1 o più)
    - Titolo TESTO
    - Abstract TESTO
    - TipologiaPub ELEMENT (DISGIUNTO)
      - RIVISTA ELEMENTO
        - CONFERENZA ELEMENTO
        - LIBRO ELEMENTO
        - RIVISTA
          - » NOME_RIVISTA TESTO
          - » IMPACT FACTOR TESTO
          - » VOLUME
        - CONFERENZA
          - » TITOLO CONFERENZA TESTO
          - » LUOGO CONFERENZA TESTO
          - » DATA TESTO
        - LIBRO
          - » EDITORE TESTO
          - » DATA PUBBLICAZIONE TESTO
    - COAUTORI TESTO (0 o più)

```
<?xml version="1.0"?>
<!DOCTYPE Autore [
<!ELEMENT Autore (Nome,Cognome,pubblicazione+)>
<!ELEMENT Nome (#PCDATA)>
<!ELEMENT Cognome (#PCDATA)>
<!ELEMENT pubblicazione (abstract, tipoPub+,coautori*)>
<!ATTLIST pubblicazione titolo CDATA #REQUIRED>
<!ELEMENT abstract (#PCDATA)>
<!ELEMENT tipoPub (rivista|conferenza|libro)>
<!ELEMENT coautori (#PCDATA)>
<!ELEMENT rivista (volume)>
<!ATTLIST rivista
  nome    CDATA #REQUIRED
  impactf CDATA #IMPLIED>
<!ELEMENT volume (#PCDATA)>
<!ELEMENT conferenza(titolo,luogo,data)>
<!ELEMENT luogo (#PCDATA)>
<!ELEMENT data (#PCDATA)>
<!ELEMENT libro (editore,data)>
<!ELEMENT editore (#PCDATA)>
<!ENTITY abs SYSTEM "absract.txt">
]>
<autore>
  <nome>Filippo</nome>
  <cognome>Gialli</cognome>
  <pubblicazione
    titolo="WEB III">
    <abstract>&abs;</abstract>
    <tipoPub>
      <libro>
        <editore>APGEO</editore>
        <data>10/10/2009</data>
      </libro>
    </tipoPub>
    <coautori>Enzo Gialli</coautori>
  </pubblicazione>
</autore>
```

- Autore
  - Nome TESTO
  - Cognome TESTO
  - Pubblicazione ELEMENTO (1 o più)
    - ATTRIBUTO DI TIPO TESTO Titolo
    - Abstract TESTO
    - TipologiaPub ELEMENT (DISGIUNTO)
      - RIVISTA ELEMENTO
        - CONFERENZA ELEMENTO
        - LIBRO ELEMENTO
        - RIVISTA
          - » ATTRIBUTO OBBLIGATORI NOME_RIVISTA TESTO
          - » ATTRIBUTO OPZIONALE IMPACT FACTOR TESTO
          - » VOLUME
        - CONFERENZA
          - » TITOLO CONFERENZA TESTO
          - » LUOGO CONFERENZA TESTO
          - » DATA TESTO
        - LIBRO
          - » EDITORE TESTO
          - » DATA PUBBLICAZIONE TESTO
    - COAUTORI TESTO (0 o più)

Entità

- 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 parsed

- Sono un modo pratico per inserire stringhe nel documento facendo riferimento a una definizione separata, invece di scriverle esplicitamente.

- Sono utili nel caso ci siano caratteri non digitabili direttamente, o per espandere stringhe usate di frequente, oppure per scrivere caratteri che non sono ammessi in maniera esplicita in un contesto, perché riservati (come le virgolette o i segni '<' e '>').

- Le entità vengono dichiarate all'interno della grammatica e vengono poi utilizzate nel documento XML.
- La sintassi è la seguente:

```
<!ENTITY nome definizione>
```

- La definizione dell'entità è l'oggetto che viene inserito dal processore XML quando viene trovato il nome dell'entità

Esempio:

DTD

```
<!ENTITY scrittore "Joseph Conrad">
<!ENTITY libro “La linea d’ombra">
```

XML

```
<scrittore>&scrittore;</scrittore>
<libro>&libro;</libro>
```

Entità esterne

- Sintassi
- Oltre alle entità interne ci sono le entità esterne. Queste non sono contenute direttamente nel documento XML ma si trovano all'interno di documenti

```
<!ENTITY nome SYSTEM uri>

Es.
DTD
...
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT autore (#PCDATA)>
<!ELEMENT abstract (#PCDATA)>
...
<!ENTITY abstract SYSTEM “abstract.txt”>

XML
<abstract>&abstract;</abstract>
```

DTD: Attributi

- **Gli attributi permettono di completare gli elementi**
- **associando ad essi alcuni semplici dati aggiuntivi.**
- **Con DTD possono essere definiti usando una dichiarazione del tipo `<!ATTLIST ...>`, la struttura è la seguente:**

```
<!ATTLIST nome_elemento
  nome_attributo1 tipo_attributo1 default1
  nome_attributo2 tipo_attributo2 default2
   .
   .
   .
>
```

- I possibili valori per i default sono:

```
#REQUIRED

#IMPLIED

#FIXED valore fisso

Default
```

- Quando un attributo è dichiarato come #REQUIRED allora deve apparire obbligatoriamente nell'elemento in cui è specificato.

- Se invece è dichiarato #IMPLIED invece è opzionale.

- Attraverso la parola chiave #FIXED può essere specificato un valore fisso che deve assumere l'attributo, che, nel caso di omissione nel documento XML, viene assunto automaticamente dal parser. Nel caso in cui venga omessa la parola chiave #FIXED, ma venga indicato solo un valore di Default (ultimo caso in tabella), allora l'attributo potrà assumere anche valori diversi da quello specificato, se invece non compare nel documento il parser utilizzerà quello indicato.

```
<!DOCTYPE Corso [
<!ELEMENT Corso (Descrizione,Temi)>
<!ATTLIST Corso
  docente CDATA #REQUIRED
  titolo CDATA #REQUIRED
  locazione CDATA “Laboratorio PC Wyeth” #FIXED
>
...
```

DTD: tipi di dati per gli attributi

- Il tipo di dato può assumere i seguenti valori:

CDATA
ENTITY
ENTITIES
ID
IDREF
IDREFS
NMTOKENS
NOTATION
Enumerated

`CDATA`

valore dell'attributo di tipo testo.

`ID IDREF IDREFS`

sono utili per specificare identificatori.

- ID

  il valore da esso assunto deve essere unico all'interno dello stesso documento XML. Questo è utile quando si vuole essere sicuri dell'unicità del dato inserito nell'attributo (es. Codice fiscale o matricola di un dipendente).

- IDREF

  utili per indirizzare attributi di tipo ID definiti da qualche altra parte nel documento.

- Es.

```
<!ELEMENT Prestito (nome, cognome)>
<!ATTLIST Prestito collocazione IDREF #REQUIRED >
```

  Prestito può fare riferimento alla collocazione del libro senza dover ripetere tutto l'elemento libro.

Esempio

```
<!DOCTYPE TVSCHEDULE [

  <!ELEMENT TVSCHEDULE (CHANNEL+)>
  <!ELEMENT CHANNEL (BANNER,DAY+)>
  <!ELEMENT BANNER (#PCDATA)>
  <!ELEMENT DAY (DATE,(HOLIDAY|PROGRAMSLOT+)+)>
  <!ELEMENT HOLIDAY (#PCDATA)>
  <!ELEMENT DATE (#PCDATA)>
  <!ELEMENT PROGRAMSLOT (TIME,TITLE,DESCRIPTION?)>
  <!ELEMENT TIME (#PCDATA)>
  <!ELEMENT TITLE (#PCDATA)>
  <!ELEMENT DESCRIPTION (#PCDATA)>

  <!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
  <!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
  <!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
  <!ATTLIST TITLE RATING CDATA #IMPLIED>
  <!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
  ]>
```

Esempio

```
<!DOCTYPE NEWSPAPER [

  <!ELEMENT NEWSPAPER (ARTICLE+)>
  <!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>
  <!ELEMENT HEADLINE (#PCDATA)>
  <!ELEMENT BYLINE (#PCDATA)>
  <!ELEMENT LEAD (#PCDATA)>
  <!ELEMENT BODY (#PCDATA)>
  <!ELEMENT NOTES (#PCDATA)>

  <!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
  <!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
  <!ATTLIST ARTICLE DATE CDATA #IMPLIED>
  <!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

  <!ENTITY NEWSPAPER "Vervet Logic Times">
  <!ENTITY PUBLISHER "Vervet Logic Press">
  <!ENTITY COPYRIGHT "Copyright 1998 Vervet Logic
  Press">

  ]>
```

Esempio

```
<!DOCTYPE CATALOG [

    <!ENTITY AUTHOR "John Doe">
    <!ENTITY COMPANY "JD Power Tools, Inc.">
    <!ENTITY EMAIL "jd@jd-tools.com">

    <!ELEMENT CATALOG (PRODUCT+)>

    <!ELEMENT PRODUCT
    (SPECIFICATIONS+,OPTIONS?,PRICE+,NOTES?)>
    <!ATTLIST PRODUCT
    NAME CDATA #IMPLIED
    CATEGORY (HandTool|Table|Shop-Professional) "HandTool"
    PARTNUM CDATA #IMPLIED
    PLANT (Pittsburgh|Milwaukee|Chicago) "Chicago"
    INVENTORY (InStock|Backordered|Discontinued) "InStock">

    <!ELEMENT SPECIFICATIONS (#PCDATA)>
    <!ATTLIST SPECIFICATIONS
    WEIGHT CDATA #IMPLIED
    POWER CDATA #IMPLIED>

    <!ELEMENT OPTIONS (#PCDATA)>
    <!ATTLIST OPTIONS
    FINISH (Metal|Polished|Matte) "Matte"
    ADAPTER (Included|Optional|NotApplicable) "Included"
    CASE (HardShell|Soft|NotApplicable) "HardShell">

    <!ELEMENT PRICE (#PCDATA)>
    <!ATTLIST PRICE
    MSRP CDATA #IMPLIED
    WHOLESALE CDATA #IMPLIED
    STREET CDATA #IMPLIED
    SHIPPING CDATA #IMPLIED>

    <!ELEMENT NOTES (#PCDATA)>

    ]>
```

I namespace

- Spesso accade che alcuni nomi di elementi o attributi usati all'interno di un documento XML entrino in conflitto. Ad esempio il nome dell'elemento titolo, potrebbe indicare il titolo di un libro o di un dipinto.

- XML fornisce un utile meccanismo in grado di definire degli spazi di nomi (chiamati *namespace*) per risolvere queste ambiguità.
- Namespace o Dizionari

- Un namespace consiste di un gruppo di elementi e di nomi di attributi.

- I nomi del namespace vengono identificati utilizzando la seguente sintassi:

  `ns-prefix:local-name`

- Ad esempio potremmo distinguere i tag come <libro:titolo> e <corso:titolo>.
- Un namespace deve essere dichiarato attraverso l'attributo **xmlns** prima di poterlo utilizzare all'interno di elemento.
- Ad esempio possiamo definire il namespace *libro* nel seguente modo:

```
<biblioteca
    xmlns:libro="http://www.esempio.org/1999/libro">
    <libro:titolo>...</libro:titolo>
</biblioteca>
```

- L'attributo xmlns definisce il namespace libro identificandolo univocamente con un *URI (Uniform Resource Identifier)* che nel nostro esempio è http://www.esempio.org/1999/libro.

```
<?xml version="1.0"?>
<clienti>
  <cliente>
     <ragione_sociale>Trinacria S.P.A.</ragione_sociale>
     <partitita_iva>1234566779</partita_iva>
     <indirizzo>Pezza Grande, Zona Industriale,CT</indirizzo>
     <citta>Catania</citta>
     <telefono>44445555</telefono>
     <indirizzo>http://www.trinacria.it</indirizzo>
     <email>segreteria@trinacria.it</email>
   </cliente>
  <cliente>
    ...
  </cliente>
</clienti>
```

- Il tag radice del documento XML (nel nostro caso <clienti>) conterrà l'indicazione del dizionario.
- Supponiamo che l'azienda abbia due punti vendita dove colleziona dati relativi a clienti.
  - Catania
  - Ragusa

```
<?xml version="1.0"?>
<clienti
   xmlns:ct="http://catania.aligroup.it/Dizionario/1.0"
   xmlns:rg="http://ragusa.aligroup.it/Dizionario/1.0"
   xmlns="http://aligroup.it/Dizionario/1.0"
>
```

```
<?xml version="1.0"?>
<clienti
   xmlns:ct="http://catania.aligroup.it/Dizionario/1.0"
   xmlns:rg="http://ragusa.aligroup.it/Dizionario/1.0"
   xmlns="http://aligroup.it/Dizionario/1.0">
  <cliente>
     <ragione_sociale>Trinacria S.P.A.</ragione_sociale>
     <partitita_iva>1234566779</partita_iva>
     <ct:indirizzo>Pezza Grande, Zona
  industriale,CT</ct:indirizzo>
     <citta>Catania</citta>
     <telefono>44445555</telefono>
     <rg:indirizzo>http://www.trinacria.it</rg:indirizzo>
     <email>segreteria@trinacria.it</email>
   </cliente>
  <cliente>...</cliente>
</clienti>
```

Sintassi

```
<radice
   xmlns:prefisso="URI"
   xmlns="URI"
>
```

- URI viene utilizzato per la definizione UNIVOCE del dizionario (namespace).
- Ha la forma di un indirizzo web, ma l'obiettivo è quello di definire l'univocità del namespace.
- Non ha nessuna relazione con i reale indirizzo web.

```
<prefisso:elemento
   prefisso:attributo1=“valore”
   prefisso:attrobuto2=“valore”>
   ...
Contenuto elelemento
</prefisso:elemento>
```

Anche nella grammatica DTD possiamo fare riferimento ai namespace:

```
<!ELEMENT clienti(cliente+)>
<!ATTLIST clienti
   xmlns:ct CDATA #REQUIRED
   xmlns:rg CDATA #REQUIRED
   xmlns    CDATA #REQUIRED
>
<!ELEMENT cliente (ragione_sociale,partita_iva,ct:indirizzo,email,
    citta,rg:indirizzo,telefono)
>
<!ELEMENT ct:indirizzo (#PCDATA)>
<!ATTLIST ct:indirizzo
    ct:sedi (corso|via|piazza) #REQUIRED
>
<!ELEMENT ..
>
```