


Linguaggi ed Applicazioni multimediali

06.01- Introduction to mark-up.
06.02- classification
06.03- SGML
06.04- XML

Hypertext

Maurizio Maffi

ISTI Information Science and Technology Institute

Linguaggio di markup

un *linguaggio di markup* descrive i meccanismi di rappresentazione (strutturali, semantici o presentazionali) del testo che, utilizzando convenzioni standardizzate, sono utilizzabili su più supporti

I diversi linguaggi di markup esistenti si distinguono fondamentalmente in:

- linguaggi di markup di tipo procedurale

  indicano le procedure di trattamento del testo aggiungendo le istruzioni che devono essere eseguite per visualizzare la porzione di testo referenziata

- linguaggi di markup di tipo descrittivo.

  lasciano la scelta del tipo di rappresentazione da applicare al testo al software che di volta in volta lo riprodurrà

Linguaggio SGML

Lo Standard Generalized Markup Language (SGML), è uno standard per la descrizione logica dei documenti ed è principalmente utilizzato nei *Document Type Definition* (DTD)

L'idea centrale dello standard è un tipo di marcatura generica chiamata "marcatura descrittiva" che definisce la struttura logica dei documenti

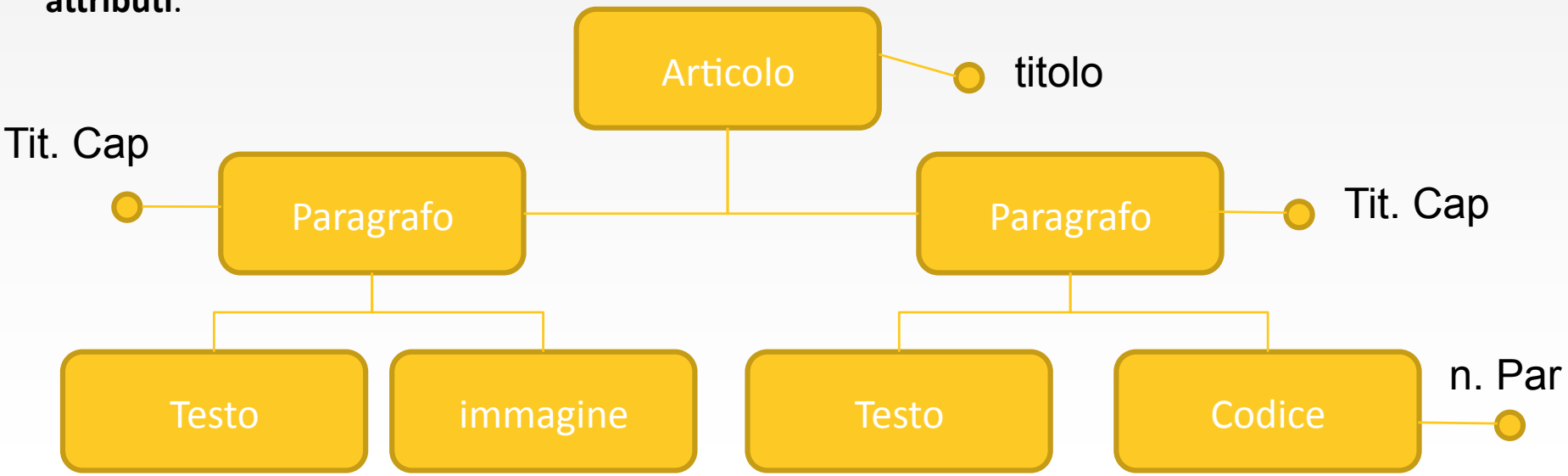
Ad esempio una lettera contiene degli elementi essenziali quali *mittente*, uno o più *destinatari*, *data*, *oggetto*, *corpo*, l'indicazione di colui che la *firma*,.... Tutti elementi che devono essere presenti, probabilmente anche con un certo ordine


Cos'è XML

- XML è un meta-linguaggio di markup, cioè un linguaggio che permette di definire altri linguaggi di markup
- XML non ha tag predefiniti e non serve per definire pagine Web né per programmare
- XML di per sé non è altro che un insieme standard di regole sintattiche per modellare la struttura di documenti e dati

Struttura dell'XML

Un documento XML è intrinsecamente caratterizzato da una **struttura gerarchica**. Esso è composto da componenti denominati **elementi**. Ciascun elemento rappresenta un componente logico del documento e può contenere altri elementi (sottoelementi) o del testo. Gli elementi possono avere associate altre informazioni che ne descrivono le proprietà. Queste informazioni sono chiamate **attributi**.



Struttura dell'XML

```
<?xml version="1.0" ?>
<articolo titolo="Titolo dell'articolo">

  <paragrafo titolo="Titolo del primo paragrafo">
    <testo>
       Blocco di testo del primo paragrafo
    </testo>

    <immagine file="immagine.jpg">
    </immagine>

  </paragrafo>

  <paragrafo titolo="Titolo del secondo paragrafo">
    <testo>
       Blocco di testo del secondo paragrafo
    </testo>

    <codice>
       Esempio di codice
    </codice>

    <testo>
       Altro blocco di testo
    </testo>
  </paragrafo>
</articolo>
```

Struttura dell'XML

```
<?xml version="1.0" ?>
```

La prima riga del documento lo identifica come un documento XML e ne specifica la versione (in questo caso la 1.0):

Il corpo vero e proprio del documento segue questa prima riga, rappresentando gli elementi tramite tag, cioè sequenze di caratteri delimitate dai segni '<' e '>' proprio come avviene per l'HTML

A differenza dell'HTML in cui i tag sono predefiniti, XML ci lascia liberi di **definire i tag** che vogliamo

Apertura e chiusura tag

Esempio Corretto:

```
<libro titolo="Divina commedia">
        <Autore>
                Dante
        </Autore>
        <immagine file="copertina.jpg" />
</libro>
```

Esempio Errato:

```
<libro titolo="Divina commedia">
        <Autore>
                Dante
        </Autore>
        <immagine file="immagine1.jpg" >
?                                        ?
```

Documenti ben formati

XML richiede un certo **rigore** sugli aspetti sintattici tutti i documenti devono essere ben formati (well formed). Perché un documento XML sia ben formato deve rispettare le seguenti regole:

- Ogni documento XML deve contenere **un unico elemento di massimo livello** (root) che contenga tutti gli altri elementi del documento. Le sole parti di XML che possono stare all'esterno di questo elemento sono i commenti e le direttive di elaborazione (per esempio, la dichiarazione della versione di XML)

- Ogni elemento deve avere un **tag di chiusura** o, se vuoti, possono prevedere la forma abbreviata (/>)

- Gli elementi devono essere opportunamente nidificati, cioè i **tag di chiusura devono seguire l'ordine inverso** dei rispettivi tag di apertura

- XML fa **distinzione tra maiuscole e minuscole**, per cui i nomi dei tag e degli attributi devono coincidere nei tag di apertura e chiusura anche in relazione a questo aspetto

- I valori degli **attributi** devono sempre essere racchiusi tra singoli o doppi **apici**

Documenti ben formati

Anche la **scelta dei nomi** dei tag deve seguire alcune regole, sono ammessi Tag che iniziano con:

- Lettere
- Underscore (_)

Le parole possono contenere i seguenti caratteri:

- A-Z a-z
- 1-0
- (.) (_) (-)

Non sono ammessi

- Spazi
- Caratteri non citati in precedenza


Documenti ben formati

Per quanto riguarda il contenuto, normalmente vengono accettati come caratteri validi in un documento XML i primi 128 caratteri della codifica ASCII

Se un documento contiene caratteri che non rientrano tra questi (es.: lettere accentate, simboli di valuta, ecc.) è necessario specificare lo schema di codifica utilizzato

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

Le specifiche di XML prevedono esplicitamente la possibilità di utilizzare la codifica Unicode per rappresentare anche caratteri non latini, come ad esempio i caratteri greci, cirillici, gli ideogrammi cinesi e giapponesi

I commenti

In XML possiamo trovare i **commenti**, cioè informazioni rivolte agli esseri umani ed **ignorate** dai software che lo elaborano. I commenti XML seguono la stessa sintassi dell'HTML, sono cioè racchiusi tra le sequenze di caratteri <!-- e **-->** e possono trovarsi in qualsiasi punto del documento.

```
<libro titolo="Divina commedia"> <!– libro disponibile -->
        <Autore>
                Dante
        </Autore>
        <immagine file="copertina.jpg" />
  </libro>
```

Caratteri particolari

L'inserimento di alcuni caratteri potrebbe non rendere il documento Well Formed.
Esempio: se dobbiamo inserire del testo che contiene il simbolo <, corriamo il rischio che possa venire interpretato come l'inizio di un nuovo tag

```
<libro titolo="matematica">
        <testo>
           possiamo affermare che 10 < 20
        </testo>
   </libro>
```

Per evitare situazioni di questo tipo, XML prevede degli oggetti speciali detti **entità** che consentono di sostituire altri caratteri

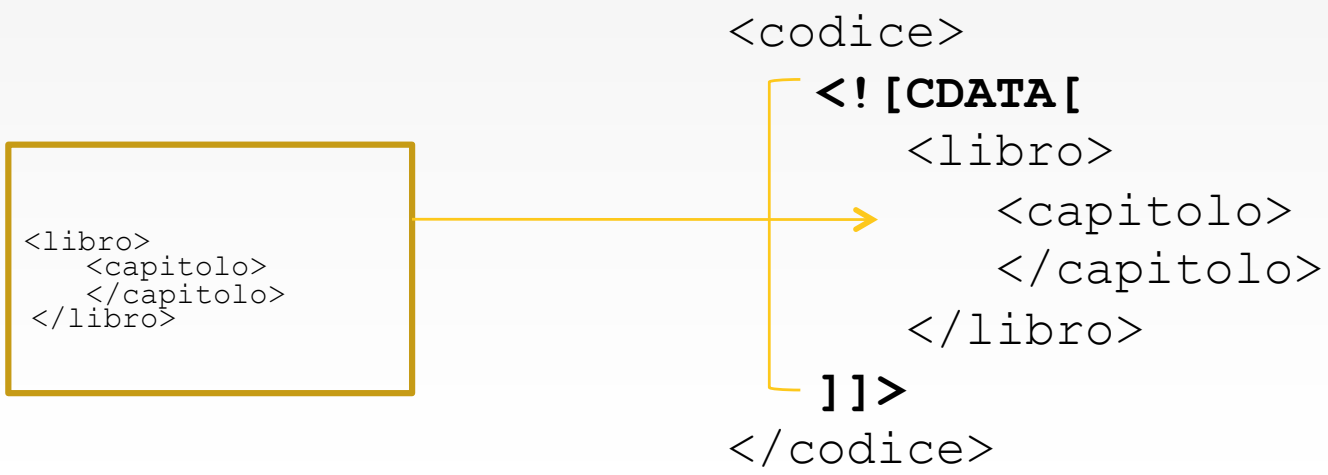
- **&** definisce il carattere &
- **<** definisce il carattere <
- **>** definisce il carattere >
- **"** definisce il carattere "
- **'** definisce il carattere '

```
<testo>
   possiamo affermare che 10 &lt 20
</testo>
```

CDATA

In determinate situazioni gli elementi da sostituire con le entità possono essere molti, il che rischia di rendere illeggibile il testo ad essere umano. Si consideri il caso in cui un blocco di testo illustri proprio del codice XML

Esempio Vogliamo scrivere nel tag codice il seguente listato:




Document Type Definition

Lo scopo di un **Document Type Definition** è quello di definire le componenti ammesse nella costruzione di un documento XML

- Definisce gli elementi leciti all'interno del documento. Non si possono usare altri elementi se non quelli definiti. Una specie di "vocabolario" per i file che lo useranno.

- Definisce la struttura di ogni elemento. La struttura indica cosa può contenere ciascun elemento, l'ordine, la quantità di elementi che possono comparire e se sono opzionali o obbligatori. Una specie di "grammatica".

- Dichiara una serie di attributi per ogni elemento e che valori possono o devono assumere questi attributi.

- Fornisce infine alcuni meccanismi per semplificare la gestione del documento, come la possibilità di dichiarare entity e la possibilità di importare parti di altri DTD


Document Type Definition (DTD).

La **sintassi di un Dtd** si basa principalmente sulla presenza di due dichiarazioni

<!ELEMENT> : definisce gli elementi utilizzabili nel documento e la struttura del documento stesso

<!ATTLIST> definisce la lista di attributi per ciascun elemento

<!ELEMENT>

```
<!ELEMENT articolo(paragrafo+)>
```

<articolo> ha come sottoelemento uno o più elementi <paragrafo>.

‘**+**’ : indica il relativo numero di occorrenze

Un insieme di caratteri speciali ha appunto lo scopo di indicare il **numero di occorrenze** di un elemento

‘**+**’ : l'elemento è presente **una o più** volte

‘*****’ : l'elemento è presente **zero o più** volte

‘**?**’ : l'elemento è presente **zero o una** sola volta

<!ELEMENT>

```
<!ELEMENT paragrafo(immagine*, testo+)>
```

<paragrafo> contiene la sequenza di elementi <immagine> e <testo>. L'elemento <immagine> può essere presente zero o più volte, mentre <testo> deve essere presente almeno una volta

Per la definizione dei tag che non contengono sottoelementi dobbiamo distinguere il caso dei tag vuoti dai tag che racchiudono testo

```
Immagine -------->> <!ELEMENT immagine empty>

Tesro        ------->> <!ELEMENT testo (#PCDATA)>
```

`#PCDATA:` che indica del testo generico

<!ELEMENT>

Esiste la possibilità di definire elementi il cui contenuto non è definito a priori, possono cioè essere vuoti o contenere altri elementi senza vincoli particolari

```
<!ELEMENT elemento ANY>
```

```
<?xml version = "1.0"?>
<!DOCTYPE nota[ <!ELEMENT nota ANY>
<!ELEMENT numero EMPTY> <!ELEMENT messaggio (#PCDATA)> <!
ELEMENT data EMPTY> ]>
   <nota>
     <numero />
        <messaggio>
            HELLO WORLD!
        </messaggio>
        <data />
   </nota>
```

Attributi <!Attlist>

Valori predefiniti degli attributi

- **#REQUIRED** Specifica che l'attributo è obbligatorio
- **#FIXED** Fornisce una dichiarazione di costante per il valore di un attributo. Se il valore è diverso da quello dichiarato, il documento non è valido
- **#IMPLIED** L'attributo è facoltativo. Cioè, se l'attributo non appare nell'elemento, l'applicazione di elaborazione può usare qualsiasi valore (se necessario).

```
<?xml version = "1.0"?> <!DOCTYPE nota [ <!ELEMENT nota (messaggio)>
<!ELEMENT messaggio (#PCDATA)>
<!ATTLIST messaggio numero CDATA #REQUIRED data CDATA #REQUIRED>]>
  <nota>
    <messaggio numero="10" data="140305">
       Hello !!
    </messaggio>
 </nota>
```

La parola chiave CDATA consente l'inclusione nella stringa di qualsiasi carattere, fuorché "<, >, &".

Attributi enumerati

Gli attributi di tipo enumerato descrivono un elenco di valori possibili per l'attributo valutato. Perché sia soddisfatto il requisito della validità, l'attributo deve avere uno dei valori presenti nell'elenco; in ogni altro caso viene considerato non valido. I valori enumerati sono separati da un carattere "pipe" (|), che è interpretato come "or" logico dal processore XML

```
<!ATTLIST messaggio Urgenza( Bassa | Media| Alta)
“bassa”>
```

Indicatori di occorrenza

Nella DTD vengono utilizzati dei simboli che predispongono il parsing a contare le occorrenze di un oggetto.

1. **,** (a, b, c) Questo operatore di sequenza separa i membri di una lista che richiede l'uso sequenziale di tutti i membri della lista (a seguito da b, seguito da c)
2. **|** (a|b|c) Questo è un operatore di scelta, che separa membri di una lista quando è richiesto l'uso di uno e uno solo dei membri (*a* o *b* o *c*).

La mancanza di un simbolo indica una occorrenza necessaria (uno e uno solo di dati).

1. **?** oggetto? Questo simbolo designa una occorrenza facoltativa (zero o un oggetto).
2. **+** paragrafo+ Questo simbolo indica un'occorrenza obbligatoria e ripetibile (almeno un paragrafo, possibilmente più d'uno).
3. ***** fratelli* Questo simbolo indica un'occorrenza facoltativa e ripetibile (zero, uno o più fratelli).


Entità

Più in generale, una entità consente di sostituire sequenze di caratteri con **nomi speciali** della forma & nome;. È possibile definire entità personalizzate all'interno di un Dtd in modo da sostituire qualsiasi sequenza di caratteri

Per definire un'entità personalizzata si utilizza la dichiarazione **<!ENTITY>**.

```
<!ENTITY html "HyperText Markup Language">
```

Grazie a questa dichiarazione possiamo utilizzare l'entità &html; al posto dell'intera stringa all'interno del documento XML che fa riferimento a questa grammatica

Indicare il Dtd

Esistono **due modi** per indicare il Dtd cui un documento XML fa riferimento.

Il primo modo prevede la presenza del Dtd all'interno del documento XML, come nel seguente esempio:

```
<?xml version="1.0">
<!DOCTYPE articolo[...Definizioni del Dtd…]>
<articolo>
    ... Content of the document ...
</articolo>
```

La dichiarazione **<!DOCTYPE>** indica che il documento individuato dall'elemento root <articolo> segue le regole definite tra le parentesi quadre

Indicare il Dtd

Il secondo modo prevede che il Dtd sia definito in un file esterno ed il documento XML abbia un riferimento a tale file, come nel seguente esempio

```
<?xml version="1.0">
<!DOCTYPE articolo SYSTEM "articolo.dtd">
```

In questo caso si fa riferimento all'URI del Dtd definito nel file articolo.dtd.

```
<!DOCTYPE articolo SYSTEM "http://www.myXML.it/articolo.dtd">
```

se il Dtd viene pubblicato su un sito web è possibile specificare il riferimento al Dtd tramite il suo URL


Problemi del DTD

l'uso dei Dtd per definire la grammatica di un linguaggio di markup **non sempre è del tutto soddisfacente**

I Dtd **NON** consentono di

• specificare un tipo di dato per il valore degli attributi

• specificare il numero minimo o massimo di occorrenze di un tag in un documento

Queste limitazioni hanno spinto alla definizione di approcci alternativi per definire grammatiche per documenti XML. tra questi approcci il più noto è **XML Schema**.


XML Schema

XML Schema è una descrizione formale di una grammatica per un linguaggio di markup basato su XML

A differenza di un Dtd, che utilizza una propria sintassi specifica, un XML Schema utilizza **la stessa sintassi XML** per definire la grammatica di un linguaggio di markup

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

... Definizione della grammatica ...

</xs:schema>
```


XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

indica al parser che in questo documento saranno utilizzati dei tag definiti dal namespace standard del W3C

XML Schema prevede il tag **<xs:element>** per la definizione degli elementi utilizzabili in un documento XML, specificando nell'attributo name il nome del relativo tag

<xs:element name="testo" type="xs:string" />

<!ELEMENT testo (#PCDATA)>

Tipi di dato semplice

XML Schema introduce il concetto di **tipo di dato semplice** per definire gli elementi che non possono contenere altri elementi e non prevedono attributi

xs:string	**Stringa di caratteri**
xs:integer	**Numero intero**
xs:decimal	**Numero decimale**
xs:boolean	**Valore booleano**
xs:date	**Data**
xs:time	**Ora**
xs:uriReference	**URL**

```
<xs:element name="quantita" type="xs:integer" />
```


Tipi di dato semplice

XML Schema prevede anche la possibilità di definire **tipi di dato semplici personalizzati** come derivazione di quelli predefiniti

```
<xs:element name="quantita" >

   <xs:simpleType>
      <xs:restriction base="xs:integer">
         <xs:minInclusive value="1" />
         <xs:maxInclusive value="100" />
      </xs:restriction>
   </xs:simpleType>

</xs:element>
```

<quantita> è di tipo semplice e prevede una restrizione sul tipo di dato intero predefinito accettando valori compresi tra 1 e 100.


Tipi di dato complesso

Lo schema generale per la definizione di un **elemento di tipo complesso** è il seguente:

```
<xs:element name="NOME_ELEMENTO">
   <xs:complexType>
      ... Definizione del tipo complesso ...
      ... Definizione degli attributi ...
   </xs:complexType>
</xs:element>
```

Se l'elemento può contenere altri elementi possiamo definire la sequenza di elementi che possono stare al suo interno utilizzando uno dei **costruttori di tipi complessi** previsti:

- **<xs:sequence>** Consente di definire una sequenza ordinata di sottoelementi
- **<xs:choice>** Consente di definire un elenco di sottoelementi alternativi
- **<xs:all>** Consente di definire una sequenza non ordinata di sottoelementi

Tipi di dato complesso

Per ciascuno di questi costruttori e per ciascun elemento è possibile definire il numero di occorrenze previste utilizzando gli attributi:

minOccurs: numero minimo di occorrenze

maxOccurs : numero massimo di occorrenze

```
<xs:element name="paragrafo">
   <xs:complexType>
      <xs:element name="testo" minOccurs="1"
      maxOccurs="unbounded"/>
   </xs:complexType>
</xs:element>
```

In questo caso il valore **unbounded** indica che non è stabilito un numero massimo di elementi testo che possono stare all'interno di un paragrafo

Tipi di dato complesso

La definizione degli attributi è basata sull'uso del tag **<xs:attribute>**, come nel seguente esempio:

```
<xs:attribute name="titolo" type="xs:string"
use="required" />
```

Documento XML

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="articolo">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="paragrafo" maxOccurs="unbounded">
     <xs:complexType>
      <xs:all maxOccurs="unbounded">
       <xs:element name="immagine" minOccurs="0">
        <xs:complexType>
         <xs:attribute name="file" use="required">
          <xs:simpleType>
           <xs:restriction base="xs:string"/>
          </xs:simpleType>
         </xs:attribute>
        </xs:complexType>
       </xs:element>
       <xs:element name="testo"/>
       <xs:element name="codice" minOccurs="0"/>
      </xs:all>
      <xs:attribute name="titolo" type="xs:string"use="optional"/>
```

Documento XML

```
<xs:attribute name="tipo" use="optional">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:enumeration value="abstract"/>
         <xs:enumeration
         value="bibliografia"/>
         <xs:enumeration value="note"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="titolo" type="xs:string"
   use="required"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```


Documento XML

Lo schema XML appena descritto è equivalente al seguente schema DTD

```
<!ELEMENT articolo(paragrafo+)>
<!ELEMENT paragrafo (immagine*, testo+, codice*)>

<!ELEMENT immagine EMPTY>
<!ELEMENT testo (#PCDATA)>
<!ELEMENT codice (#PCDATA)>

<!ATTLIST articolo titolo CDATA #REQUIRED>
<!ATTLIST paragrafo
      titolo   CDATA #IMPLIED
      tipo     (abstract|bibliografia|note) #IMPLIED
>
<!ATTLIST immagine file CDATA #REQUIRED>
```


Dichiarazione di tipi

XML Schema prevede la possibilità di rendere modulare la definizione della struttura di un documento XML tramite la dichiarazione di tipi e di elementi, ossia possiamo analizzare ciascun sottoelemento significativamente complesso e **fornire una definizione separata** come elemento o come tipo di dato

```
<xs:complexType name="nome_tipo">
...
</xs:complexType>
```

```
<xs:element name="nome_elemento" type="nome_tipo" />
```


Dichiarazione di tipi - esempio

Il seguente codice riporta lo XML Schema per un linguaggio di descrizione di articoli visto nell'esempio precedente

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="articolo">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="paragrafo" type="paragrafoType"
    maxOccurs="unbounded"/>
   </xs:sequence>
  <xs:attribute name="titolo" type="xs:string"
  use="required"/>
  </xs:complexType>
 </xs:element>
```

Dichiarazione di tipi - esempio

```
<xs:complexType name="paragrafoType">
  <xs:all maxOccurs="unbounded">
   <xs:element name="immagine" type="immagineType‟ minOccurs="0"/>
   <xs:element name="testo"/>
   <xs:element name="codice" minOccurs="0"/>
  </xs:all>
  <xs:attribute name="titolo" type="xs:string‟ use="optional"/>
  <xs:attribute name="tipo" use="optional">
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:enumeration value="abstract"/>
     <xs:enumeration value="bibliografia"/>
     <xs:enumeration value="note"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
 </xs:complexType>
```

Dichiarazione di tipi - esempio

```
 <xs:complexType name="immagineType">
  <xs:attribute name="file" use="required">
   <xs:simpleType>
    <xs:restriction base="xs:string"/>
   </xs:simpleType>
  </xs:attribute>
 </xs:complexType>
</xs:schema>
```

Integrazione di grammatiche e namespace

```
<articolo
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="articolo.xsd"
   titolo="prova" >
```

L'attributo **xmlns:xsi** indica un URL che specifica la modalità con cui si indicherà il riferimento allo schema XML. L'attributo **xsi:noNamespaceSchemaLocation** indica il nome e l'eventuale percorso del file contenente lo schema XML di riferimento


Presentazione di XML con CSS

Un metodo per gestire la **presentazione del contenuto** di un documento XML consiste nell'utilizzare i Cascading Style Sheets (CSS).

I fogli di stile CSS sono pensati principalmente per il Web e mancano pertanto di alcune caratteristiche che possono risultare utili in ambiti diversi

Tra le principali limitazioni, non è prevista la possibilità di estrarre il valore degli attributi degli elementi in modo da poterli visualizzare


XSL: eXtensible Stylesheet Language

L'eXtensible Stylesheet Language (XSL) è un insieme di tre linguaggi che forniscono gli strumenti per l'elaborazione e la presentazione di documenti XML in maniera molto flessibile

- **XPath** consente di individuare gli elementi e gli attributi di un documento XML sui quali verranno applicate le operazioni necessarie per la presentazione dei dati
- **XSLT** (XSL transformation) consente di controllare le operazioni che rendono i dati presentabili
- **XSL-FO** (XSL Formatting Objects) definisce un insieme di tag di formattazione

Questa **suddivisione dei compiti** nel processo di presentazione è il punto di forza di XSL e ne garantisce la flessibilità. Infatti, questi tre linguaggi non sono strettamente dipendenti l'uno dall'altro