XML e TEI: introduzione alla codifica dei testi letterari

Con la codifica dei testi si intende la rappresentazione dei testi stessi su un supporto digitale in un formato utilizzabile dall'elaboratore (**M**achine **R**eadable **F**orm) mediante un opportuno linguaggio teorico.

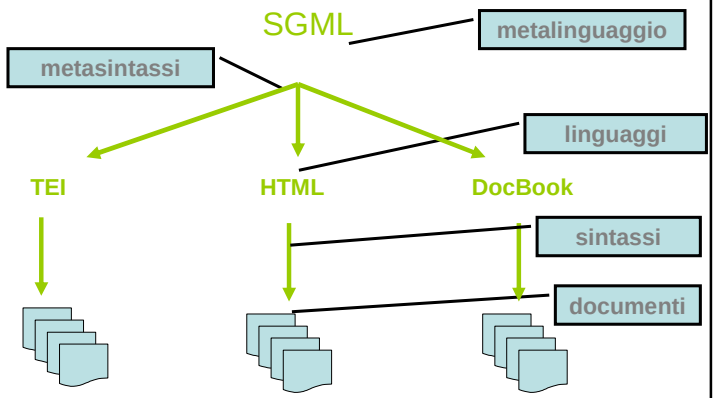
Tale linguaggio andrà a descrivere il testo in ogni sua parte attraverso delle **marche** o **tag**, ovvero stringhe di caratteri definite dalle due parentesi uncinate **< >**, al cui interno sono specificati dei comandi.

La nascita del linguaggio

Il progetto XML ha avuto inizio alla fine del 1996, nell'ambito della *SGML Activity* del W3C. L'interesse che questo progetto, il più importante dell'organizzazione dopo il World Wide Web, ha suscitato, ha condotto il W3C a creare un gruppo di lavoro, *XML Working Group*, composto da esperti mondiali delle tecnologie SGML, ed una commissione, *XML Editorial Review Board*, deputata alla redazione delle specifiche del progetto. Nel Febbraio del **1998**, le specifiche sono divenute una raccomandazione ufficiale, con il nome *Extensible Mark-up Language* (XML) versione 1.0.

- L'**XML** non è soltanto uno strumento per il WEB ma è qualcosa di più: è uno strumento che permetterà la condivisione totale dei dati, intesi come pura informazione, a prescindere dal tipo di visualizzazione e di utilizzo che se ne farà in futuro.
- XML è un acronimo per e**X**tensible **M**arkup **L**anguage: Language perché si tratta di un linguaggio, Markup perché è fondato sull'utilizzo dei marcatori ed eXtensible perché consente a chi lo utilizza di creare i *tag* di cui ha bisogno.
- **Metalinguaggio**. Non set di marcatori predefinito come HTML (cfr. DTD) ma istruzioni di SINTASSI per la creazione di molteplici linguaggi di codifica.

Il concetto di metalinguaggio di codifica

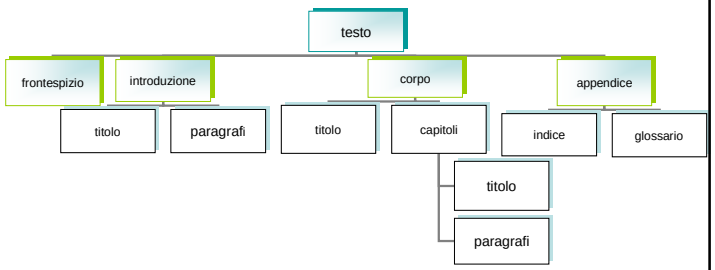


- Le raccomandazioni del W3C sull'XML sono:
  - **XML dovrà supportare un largo campo di applicazioni (motori per la visualizzazione di contenuti, strumenti di traduzione e applicazioni di database);**
  - **XML dovrà essere compatibile con SGML**
  - **XML dovrà essere facilmente interpretabile in modo da facilitarne la diffusione;**
  - **XML non dovrà avere opzioni perché possono dare problemi di compatibilità;**
  - **XML dovrà essere leggibile dall'uomo anche se questi non ha un Parser XML ma un semplice editor;**
  - **XML dovrà avere una progettazione formale e concisa non come SGML;**
  - **I documenti XML dovranno essere facili da creare anche con un semplice editor.**

Concetti di base

- Il rapporto struttura logica e layout: XML codifica la STRUTTURA del documento ma non si preoccupa di come apparirà sullo schermo.
- Documenti VALIDI e BEN FORMATI. XML consente di codificare documenti usando marcatori a discrezione del codificatore, scelti sulla base della STRUTTURA del documento (BEN FORMATO - well formed) oppure di utilizzare una DTD (esistente o creata ad hoc per il tipo di documento - VALIDO).
- Lavorare con l'XML: il documento, la DTD (Document Type Definition - opzionale), il foglio di stile (XSL - eXtensible Style Sheet Language).

Struttura ad albero di un documento codificato in XML



Il Markup: Elementi, attributi, entità

- Markup come sistema di descrizione dei documenti; usa la convenzione dei delimitatori (apertura <nome>; chiusura</nome>) per racchiudere l'informazione sul testo.
- ELEMENTO: caratteristiche della partizione logica della sezione di testo
  (es.<nomepersona>Francesca</nomepersona>)
- ATTRIBUTO: caratteristiche specifiche dell'elemento
  (es.<nomepersona tipo="f">Francesca</nomepersona>)
- ENTITA': riferimento ad oggetti "esterni" al documento. Convenzione &nomeentità;

La sintassi dell'XML

- No elementi vuoti <img>, nuova sintassi: <img></img> oppure <img/>
- Case sensitive: <IMG></IMG> ma non <img></IMG>
- Valore dell'attributo fra virgolette
- Corretta nidificazione: NO MA
- i simboli "**< >**" vanno usati solo per includere i comandi dei *tag*, il simbolo **&** deve essere usato come riferimento per le entità (**á**), dove per entità si intende una sequenze arbitraria di byte a cui viene dato un nome mediante la dichiarazione della DTD.

Le Entità predefinite

- Il segno 'et' (&), le virgolette doppie e singole (' e ") e i segni dei tag (< e >) non possono apparire nel testo del documento. La ragione è che questi cinque caratteri sono riservati per le istruzione di processo di XML. Volendo utilizzare questi caratteri devono essere sostituiti con i cosiddetti riferimenti di entità (entity references) e in questo modo non vengono interpretati come parti del markup. **Un riferimento di entità è la combinazione di vari caratteri scritti fra un & e un punto e virgola.**
- I cinque riferimenti di entità predefiniti in XML sono:
- & = &
- < = <
- >= >
- " = "
- ' = '

Editor, parser e browser

- Scrivere un documento XML: editor di testo (come Blocco Note di Windows) e software visuali (come Xmetal, TextPad, XML Pro, etc.)
- Analizzare la correttezza del documento: parser validanti (verificano che il documento segua le regole sintattiche del linguaggio e le norme specificate nella DTD) e non validanti (verificano solo l'adeguatezza sintattica del documento)
- La distribuzione sul Web: Internet Explorer 5 (con parser integrato)

Un semplice documento XML

Essendo come l'HTML un formato "solo testo" è possibile realizzare il documento partendo da un editor di testo qualsiasi (come "Blocco Note" di Windows).

```
<catalogo>
    <libro numero="1">
        <autore>Cesare Pavese</autore>
        <titolo>La casa in collina</titolo>
    </libro>
    <br/>
    <libro numero="2">
        <autore>Francesco Petrarca</autore>
        <titolo>Il Canzoniere</titolo>
    </libro>
</catalogo>
```

Codifica del primo sonetto del *Canzoniere* di Petrarca:

root element

```
<antologia>
 <poesia><titolo>I</titolo>
  <stanza>
   <verso>Voi ch'ascoltate in rime sparse il suono</verso>
   <verso>di quei sospiri ond'io nudriva 'l core</verso>
   <verso>in sul mio primo giovenile errore</verso>
   <verso>quand'era in parte altr'uom da quel ch'i' sono,</verso>
  </stanza>
  <stanza>
   <verso>del vario stile in ch'io piango et ragiono</verso>
   <verso>fra le vane speranze e 'l van dolore</verso>
   <verso>ove sia chi per prova intenda amore</verso>
   <verso>spero trovar pietà, nonché perdono.</verso>
  </stanza>
<!-- altre stanze -->
 </poesia>
<!-- altre poesie -->
</antologia>
```

commento

Struttura di un documento XML

- Dichiarazione XML
- Doctype Declaration – DTD esterna (opzionale)
- Eventuali dichiarazione di elementi che compongono i documenti – DTD interna
- Eventuali riferimenti a file esterni o sostituzione di stringhe (entità)
- Eventuali riferimenti a file ancillari (fogli di stile, collezioni di entità)
- Una istanza di documento

Struttura di un documento XML

- Ad es.:
- <?xml version="1.0" encoding="iso-8859-1"?>
- <!DOCTYPE antologia SYSTEM "antologia.dtd" [
- <!ENTITY antologia2 SYSTEM "antologia2.xml">
-]>
- <antologia>
- <! -- codifica -- >
- &antologia2;
- </antologia>

dichiarazione

Dtd esterna

Entità esterna

Il Prologo e l'istanza

- **1. Prologo** : Dichiarazione obbligatoria + set di caratteri utilizzato (opzionale) (UTF-8 o ISO-8859-1)

Dichiarazione XML (case-sensitive!)
<?xml version="1.0" encoding="ISO-8859-1"?>

- **2**. eventuale DTD cui fare riferimento:

<!DOCTYPE nomeelementoradice SYSTEM "nomedtd.dtd">

(vedi per l'es. precedente il collegamento ad una dtd creata ad hoc collocata nella stessa cartella del file xml)
<!DOCTYPE antologia SYSTEM "antologia.dtd">

- **3. Istanza del documento**: Elemento **radice** e serie dei tag in struttura gerarchica.

La DTD
Document Type Definition

La DTD raccoglie l'elenco dei marcatori utilizzabili in fase di codifica, gli attributi eventuali e definisce le relazioni fra gli elementi. Esistono DTD già fatte ma è possibile anche crearne di nuove. Usare una DTD di riferimento in XML non è obbligatorio. Se il documento XML fa riferimento ad una DTD si dice VALIDO; se il set di marcatori è d'invenzione il documento è solo BEN FORMATO cioè rispetta le regole sintattiche del linguaggio.

Inserire la DTD nel prologo

Es Dtd interna al file xml

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nomeelementoradice [
<!ELEMENT nomeelementoradice (contenuto)>
]>
< nomeelementoradice >
….
</ nomeelementoradice >

Una DTD comincia con <!DOCTYPE e termina con]>. Questo segnala al processore XML dove la DTD inizia e finisce. Direttamente dopo il <!DOCTYPE viene il nome dell'elemento radice seguito da un [.

Richiamare la DTD nel prologo

<!DOCTYPE nomelementoradice SYSTEM "nomedtd.dtd">
DTD ad uso privato, generalmente situata nell'hard disk (e in questo esempio deve essere anche nella stessa cartella)

<!DOCTYPE nomelementoradice PUBLIC nomeconvenzionale "URL/nomedtd.dtd">
DTD ad uso pubblico. In questo casi alla DTD viene assegnato un nome univoco che l'elaboratore XML cerca per procurarsi la DTD (oppure utilizza l'URL spcificato).

Es: **<!DOCTYPE TEI.2 PUBLIC "-//TEI P3//DTD Main Document Type//EN">**
In questo caso indica che il documento è della TEI.2, che la DTD è stata sviluppata dalla TEI P3 e che la lingua utilizzata è l'inglese (EN).

Che cos'è la DTD

Definisce il tipo di *elemento* ed il *content model* (indica quali elementi possono occorrere all'interno dell'elemento stesso, in che ordine e con quale ricorrenza)

<!ELEMENT testo (intro, corpo, app)>

<!ELEMENT Generic Identifier (content model) >

Che cos'è la DTD

In questo caso l'elemento definito è *testo* ed all'interno delle parentesi tonde è espresso il *content model*. Quest'ultimo ha una sintassi molto complessa, composta da **indicatori di occorrenza** e **connettivi**.

Es. di una dtd ad hoc

(vedi es.precedente)

- <!ELEMENT antologia (poesia+)>
- <!ELEMENT poesia (titolo?, stanza+)>
- <!ELEMENT titolo (#PCDATA)>
- <!ELEMENT stanza (verso+)>
- <!ELEMENT verso (#PCDATA)>

#PCDATA: si tratta dell'abbreviazione di **parsed character data**, e significa che l'elemento che si sta dichiarando può contenere qualsiasi carattere valido (ma non elementi).

Indicatori di occorrenza

- Indicatori di occorrenza indicano quante volte l'*elemento* nominato nel *content model* può essere può essere utilizzato.
- "*" zero o più ricorrenze;
- "?" zero o una ricorrenza;
- "+" una o più ricorrenze.

Es. <!ELEMENT poesia (titolo?, stanza+)>

La dichiarazione di <poesia> (titolo?, stanza+) indica che possiamo avere delle <poesia> con un <titolo> oppure senza, ma che ci deve essere almeno una <stanza>.

La virgola indica che il titolo deve sempre precedere la stanza

Connettori

- "|" almeno uno degli elementi della lista deve comparire;
- "," entrambi gli elementi separati da questo simbolo devono comparire nell'ordine specifico.

Se sostituissimo nell'esempio precedente la virgola con una barra verticale, allora una <poesia> potrebbe essere composta da un <titolo> o da <stanza>, ma non da tutti e due!

Gli Attributi

```
<!ELEMENT poesia (titolo?, stanza+)>
<!ATTLIST poesia
        id      ID
        #IMPLIED
        status (bozza | revisionato | pubblicato) “bozza” >
```

- La dichiarazione comincia con il termine chiave **ATTLIST** che introduce le specifiche di una lista di attributi (*attribute list specification*). La prima dichiarazione specifica l’elemento di cui si sta dichiarando la lista di attributi, poesia nel nostro esempio. A questo segue una serie di righe, una per ogni attributo che deve essere dichiarato, ciascuna composta di tre parti che specificano rispettivamente :
  - il nome dell’attributo
  - il tipo di valori che può prendere
  - il valore di default

Le Entità

- Le entità possono essere di tre tipi:
- Interne (generali)
- Esterne (generali)
- Parametriche

Una volta dichiarata un’entità nella DTD, si potrà richiamare attraverso un riferimento di entità (*entity reference*). Un riferimento si utilizza racchiudendo il nome dell’entità fra ‘&’ e ‘;’; ad esempio *&xml;* secondo l’esempio precedente.

Le Entità generali interne

Le *entità interne* servono per sostituire una stringa di caratteri con un’altra. Ad esempio, potremmo decidere di codificare un entità *xml* in modo che ogni volta che noi scriviamo tale sigla, in realtà corrisponda ad *eXtensible Markup Language*

<!ENTITY nome “stringa da sostituire”>

Nel documento la sostituzione avverrà tutte le volte che l’elaboratore leggerà &nome;

Esempio:

<!ENTITY xml “Extensible Markup Language”> da richiamare nel documento XML con &xml;

Le Entità Generali esterne

- **Le *entità esterne* sono gruppi di dati immagazzinati in file diversi rispetto a quello principale e vengono utilizzate per spezzare grossi file, rendendoli più maneggevoli; possiamo richiamare file xml o di puro testo (analizzabili dal parser)**

<!ENTITY antologia2 SYSTEM “antologia2.xml”>

Il file viene recuperato dal sistema tutte le volte che trova la stringa **&antologia2;**

Con le entità si può proteggere dati non XML (non analizzabili) dagli analizzatori sintattici, così da includerli in documenti con dei riferimenti.

Sezioni CDATA e Commenti

- CDATA sono delle sezioni di testo che il parser XML non cerca di interpretare. Tutte le occorrenze di & in una sezione CDATA, per esempio, verranno letti dal parser come & e un simbolo < non sarà interpretato come istruzione di markup. Le sezioni CDATA s'iniziano con <![CDATA[e terminano in]]>.
- Il commento è una sezione speciale che comincia con <!-- e finisce con --> . Tutti i dati scritti fra questi due tag sono ignorati dal processore XML. I commenti sono per lo più utilizzati per aggiungere brevi note nel documento XML, o per commentare intere sezioni del documento XML.

Il foglio di stile

XML codifica la STRUTTURA del documento e delega ad altri linguaggi il compito di definire quale sarà invece il layout, cioè l'aspetto, la formattazione del documento, come cioè il testo codificato apparirà sullo schermo in fase di visualizzazione.

Il foglio di stile contiene dunque le istruzione di formattazione: ogni porzione di testo che sta tra due marcatori assumerà l'aspetto definito per quel marcatore nel foglio di stile

CSS e XSL

XML per la formattazione usa i CSS (Cascading Style Sheet). Ma è anche nato un nuovo linguaggio ad hoc per l'XML che si chiama **XSL** (eXtensible Stylesheet Language). E' un vocabolario di elementi con regole di formattazione.

Si basa sulle regole di formattazione dell'HTML e su quelle dei CSS. Prevede l'assegnazione di caratteristiche fisiche per ciascuno dei marcatori utilizzati nel documento XML e quindi definisce come ogni porzione di testo apparirà in fase di visualizzazione.

La visualizzazione del documento - I fogli di stile

Anche il foglio di stile è realizzabile ricorrendo ad un semplice editor di testo. Il file avrà estensione .xsl.

Un foglio di stile XSL è un documento xml che utilizza una DTD i cui elementi sono noti al motore XSLT (processore presente nei browser di ultima generazione)

Intestazione:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

L'istruzione iniziale per applicare tutta la formattazione all'intero documento:

```
<xsl:template match="/">
        <xsl:apply-templates select="nomedellaroot"/>
    </xsl:template>
```

La formattazione degli elementi con l'HTML

```
<xsl:template match="autore">
    <center>
    <font color="red" size="+3" face="Arial">
        <xsl:apply-templates/>
    </font>
    </center>
</xsl:template>
```

Il modello di costruzione: template

- **xsl:template** è la regola da applicare se l'elemento in esame corrisponde al valore dell'attributo match.
- Di volta in volta applicherò il template della radice ('*/*').
- **xsl:apply-templates** spinge a cercare, all'interno dell'elemento che stiamo considerando, se esistono altri templates applicabili. E' il modo per fare ripartire la ricerca, ricorsivamente.

Inserire immagini e creare link

```
<xsl:template match="paragrafo">
    <img src="img.jpg"/>
    <a href="link.html">
        <xsl:apply-templates/>
    </a>
</xsl:template>
```

Altre regole per i fogli di stile

```
<xsl:template match="divisione/paragrafo">
    <xsl:apply-templates/>
</xsl:template>
```

Se ho più elementi "paragrafo" che dipendono però da elementi di livello superiore diverso, posso specificare su quale degli elementi "paragrafo" voglio applicare la formattazione.

Altre regole per i fogli di stile

<xsl:template match="elemento[@attributo='valore']">

<xsl:apply-templates/>

</xsl:template>

ESEMPIO:

<xsl:template match="paragrafo[@numero='1']">

<xsl:apply-templates/>

</xsl:template>

In questo modo posso applicare la formattazione solo a quegli elementi che hanno un attributo con il valore specificato. Senza specifica del valore posso farlo con tutti gli elementi che hanno quell'attributo, indipendentemente dal valore.

<xsl:template match="elemento[@attributo]">

<xsl:apply-templates/>

</xsl:template>

Altre regole per i fogli di stile

<xsl:template match="paragrafo">

<xsl:value-of select="@attributo">

<xsl:apply-templates/>

</xsl:template>

In questo modo posso specificare che voglio visualizzare il valore dell'attributo (associato all'elemento indicato) in fase di layout.

Il documento e il foglio di stile

[Dichiarazione XML:]

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

[Collegamento al foglio di stile:]

<?xml-stylesheet href="nomedelfogliodistile.xsl" type="text/xsl"?>

<elemento radice>

[testo con marcatori]

</elemento radice>