La testualità digitale: i linguaggi di *markup* (SGML, HTML, XML)

Informatica umanistica a.a. 2018-19
Francesca Tomasi francesca.tomasi@unibo.it

La codifica dei dati elementari

La **codifica del testo** basata sulle sole **tavole dei codici**, insieme ai limiti di portabilità e compatibilità comporta un'ulteriore restrizione:

- ✓ essa consente di rappresentare nella memoria del computer solo la **sequenza dei segni grafici** che rappresentano il testo;
- ✓ un testo, sia esso a stampa o manoscritto, come noto, contiene una serie di **informazioni**, a diversi livelli, che superano la mera sequenza di caratteri.

Distingueremo la codifica di 'basso livello' o **codifica dei dati elementari** da una codifica di 'alto livello' o rappresentazione dei dati a livello di 'strutture intermedie' che chiamiamo ***markup***.

La codifica dei caratteri

- ✓ ASCII (*American Standard Code for Information Interchange* http://www.asciitable.it).
  - 7 bit 128 caratteri ($2^7 = 128$)
  - 8 bit 256 caratteri ($2^8 = 256$)

Proliferazione=difficoltà di interscambio

- ✓ ISO (*International Organization for Standardization* http://www.iso.org/iso/home.html) 8859-n (ASCII 7 bit + 1)
- ✓ UNICODE (http://www.unicode.org). 16 bit, comprende ben 65.536 caratteri (2^{16}). Ultima versione 21 bit (2^{21}) oltre 1 milione di caratteri.
  - UTF-8 (ASCII 7 bit + 1)

Ruolo e funzione dei marcatori

Con *markup* ci riferiamo alla possibilità di aggiungere alla sequenza di caratteri che rappresentano il documento digitale, **altre stringhe di caratteri** denominate **marcatori**, utili a descrivere determinati aspetti funzionali alla produzione del documento elettronico.

Questo processo di aggiunta di stringhe al flusso dei caratteri permette di **specificare determinate caratteristiche del testo**.

Siamo abituati all'uso dei *word processors* che ci consentono di effettuare questa operazione manipolando un'interfaccia grafica costituita da bottoni e menù: definire grassetti, realizzare corsivi, assegnare ad una porzione di testo uno stile, cambiare il tipo di carattere o le sue dimensioni.

Il *markup* come modellazione

La codifica dei caratteri prima e il *markup* poi rappresentano un **processo interpretativo**, risultato dell'analisi del testo.

Riguardano quindi la costruzione di un **modello** di quel testo più adeguato alle esigenze della rappresentazione elettronica.

Codificare tramite **linguaggi formali di rappresentazione del testo** significa:

- ✓ effettuare un'**analisi** del testo, mirata ad individuarne le caratteristiche ;
- ✓ formulare un'**interpretazione** della fonte, sulla base delle *features* del documento;
- ✓ in modo direttamente proporzionale agli **scopi** del trattamento informatico.

Dal *layout* alla struttura

Il termine *markup* deriva dalla **stampa tipografica tradizionale** per riferirsi a quell'insieme di **simboli e annotazioni** che l'autore o l'editore aggiunge al manoscritto per istruire lo stampatore sulle caratteristiche del documento da destinare alla stampa (p.e. 'centrato', 'titolo', 'nota', 'grassetto', 'a capo').

Funzione dei linguaggi di *markup* è di fornire un insieme di **strumenti** che consentano di aggiungere **notizie** sul testo. Tali notizie possono riguardare due differenti livelli:

- l'**aspetto**: formattazione (stili, tipi di carattere, colore e dimensione dei font, ecc.) e disposizione degli elementi nella pagina (allineamento, organizzazione spaziale delle componenti testuali, disposizione di note e annotazioni, ecc.).
- la **struttura logica** (o astratta o formale): funzione dei blocchi di testo (paragrafi, titoli, note, capitoli, ecc.), cioè organizzazione delle porzioni secondo un sistema formale identificabile.

I sistemi WYSIWYG

Word-processors: programmi che consentono all'utente di effettuare operazioni di scrittura, correzione, lettura di un qualsiasi testo, permettendo altresì la preparazione del formato dello stesso testo al fine della stampa.

I sistemi di *text processing* (cioè di manipolazione o trattamento del testo) basati sull'impiego di *word processors* sono detti di tipo *WYSISYG* (*What You See Is What You Get*) cioè: ciò che vedi (sullo schermo dell'elaboratore) è ciò che ottieni (una volta stampato il documento).

Questi sistemi agevolano notevolmente il lavoro dell'utente, consentendogli di interagire tramite l'interfaccia grafica. Permettono quindi di manipolare la rappresentazione visibile sullo schermo, e gestire le attività di formattazione e impaginazione.

Il limite dei software proprietari

Il problema sostanziale è che questi sistemi legano l'elaborazione del testo ad un determinato **programma**, rendendo quindi problematica la portabilità tra ambienti hardware e software diversi.

Si pensi al programma di videoscrittura *Word* della casa produttrice Microsoft che non solo pone limitazioni alla portabilità del formato doc da un PC a un Macintosh, ma crea anche difficoltà di lettura da una versione all'altra del programma stesso.

Impiegando cioè, al fine della rappresentazione del testo, dei **caratteri di controllo invisibili**, immessi dentro il file di testo, questi linguaggi rendono il file leggibile esclusivamente dal sistema che l'ha generato.

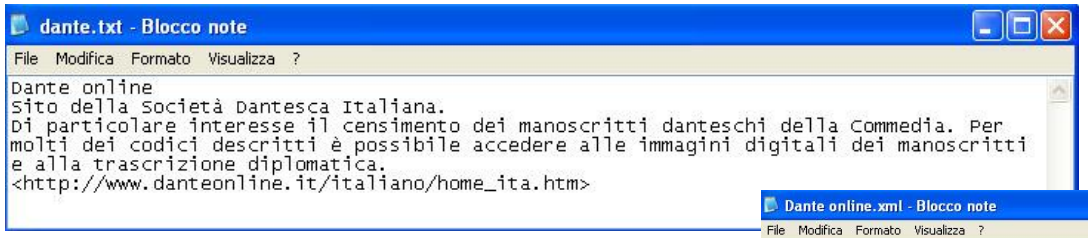
È **l'applicativo che inserisce la marcatura**, impiegando un linguaggio proprietario, che lega il prodotto finale al software di creazione.

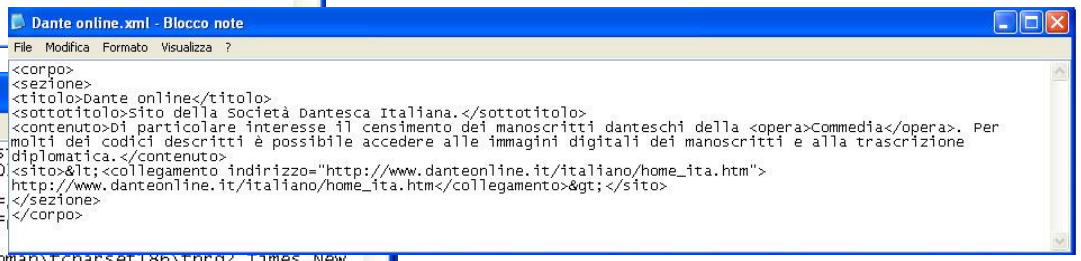
Markup languages

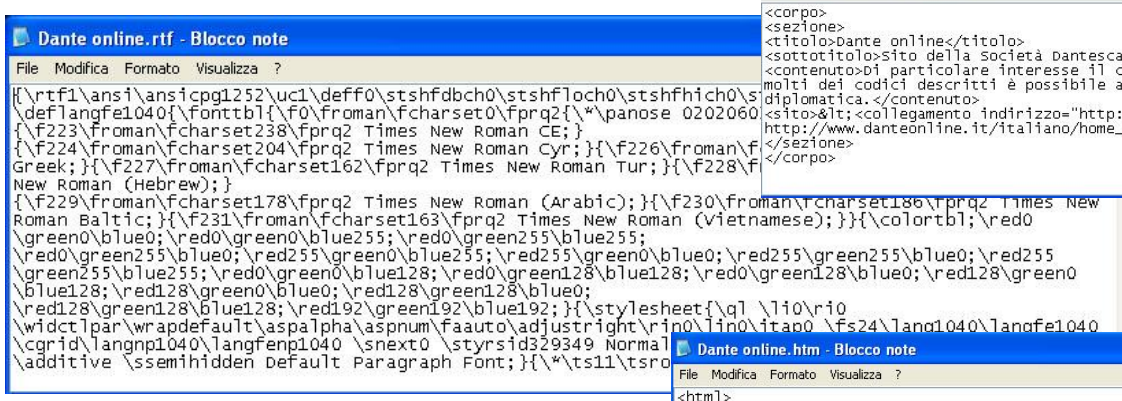
Se i *word processors* consentono una forma di *markup* del testo, si parla più propriamente di *markup languages*, detti in italiano linguaggi di marcatura del testo, per riferirsi a linguaggi che si basano su un **insieme di istruzioni e indicazioni** orientate alla **descrizione dei fenomeni di strutturazione, composizione, impaginazione** del testo stesso.

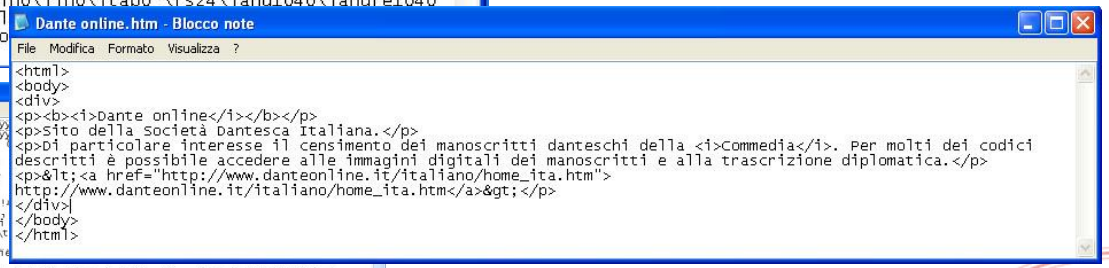
I marcatori si configurano come sequenze di caratteri visibili, e vengono immessi dentro il file, secondo una determinata **sintassi**, immediatamente accanto alla sequenza di caratteri a cui si riferiscono e cioè marcando direttamente i blocchi di testo cui intendono assegnare una determinata funzione. Garantiscono in questo modo la **leggibilità** (che non necessariamente significa però completa comprensione) del codice introdotto.

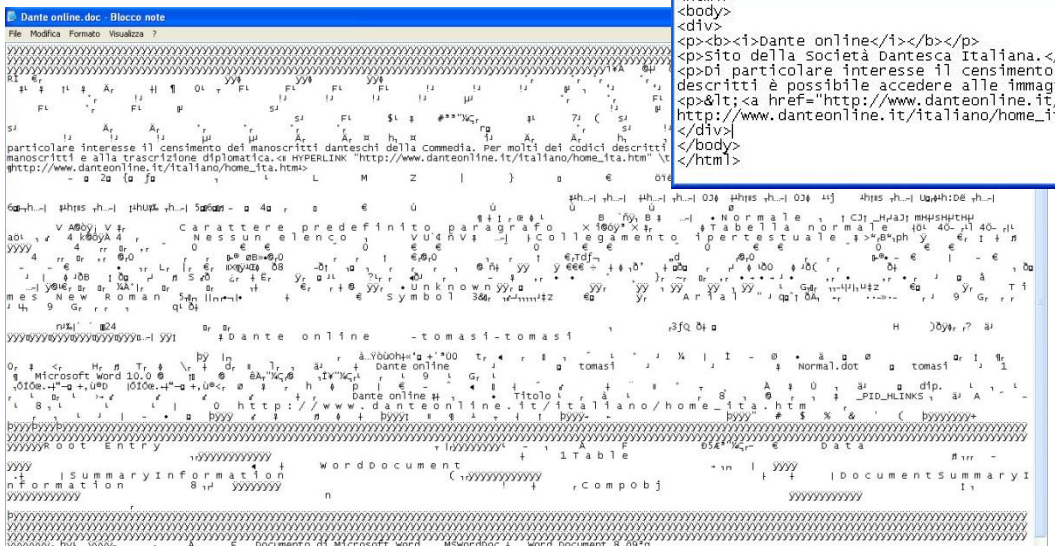
Dante online
Sito della Società Dantesca Italiana.
Di particolare interesse il censimento dei manoscritti danteschi della *Commedia*. Per molti dei codici descritti è possibile accedere alle immagini digitali dei manoscritti e alla trascrizione diplomatica.
<http://www.danteonline.it/italiano/home_ita.htm>











Markup procedurale e *markup* dichiarativo

È possibile distinguere due diverse classi di linguaggi di *markup*, che differiscono per il tipo di indicazioni impiegate all'atto dell'operazione di marcatura ovvero per la **tipologia** e la **funzione** dei marcatori utilizzati:

- ✓ *markup* specifico (o procedurale)
- ✓ *markup* generico (o dichiarativo).

Linguaggi di *markup* dichiarativo

Principali linguaggi di *markup* dichiarativo:

- ✓ SGML, il capostipite;
- ✓ HTML, una sua diretta derivazione pensata per l'implementazione di ipertesti;
- ✓ XML, un sottoinsieme semplificato dell'SGML pensato per la realizzazione di testi elettronici portabili anche sul Web e a scopi di preservazione.

I linguaggi dichiarativi:

- ✓ si basano principalmente sulla descrizione della **struttura** logica del documento e sono quindi prevalentemente utilizzati a scopo **descrittivo**;
- ✓ il formato dei dati **non è proprietario**;
- ✓ i marcatori sono **leggibili** dall'utente;
- ✓ sono *platform-independent* perche basati esclusivamente su istruzioni espresse in formato **'solo testo'**.

SGML

SGML (*Standard Generalized Markup Language*), GML prima di diventare standard, è stato elaborato nel 1986 da Charles Goldfarb con lo scopo di definire uno schema linguistico standard a livello internazionale nell'ambito della codifica dei testi e superare la moltitudine di sistemi di codifica fino ad allora sviluppati;

obiettivo primario dello standard è consentire l'**interscambio** di documenti in formato elettronico tra ambienti hardware e software differenti e garantire quindi la **portabilità** dei dati.

La struttura gerarchica ad albero

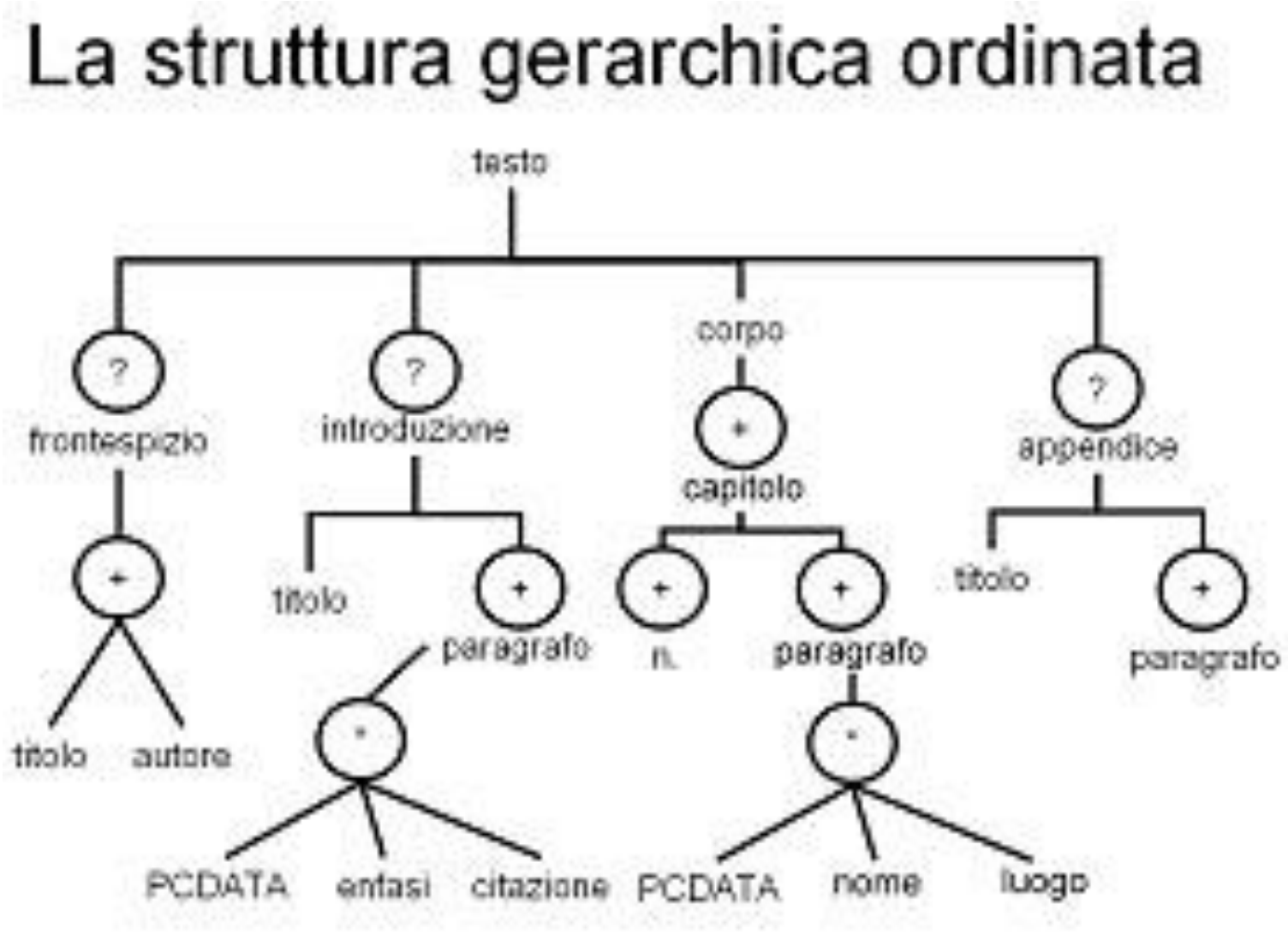
SGML si basa su di un *markup* generico, orientato alla descrizione della struttura logica di un documento di testo piuttosto che del suo aspetto.

Si fonda sull'idea che ogni documento sia dotato di una struttura astratta (o logica) definibile tramite una organizzazione rigidamente gerarchica dei suoi elementi costitutivi.

La struttura astratta di un **qualsiasi** documento viene identificata in una rappresentazione ad **albero** in cui:

- ✓ a ciascun nodo dell'albero corrisponde un elemento (e cioè ogni partizione logica della fonte);
- ✓ ai rami uscenti da ogni nodo corrispondono le relazioni tra elementi e sotto-elementi ad un dato livello (le relazioni tra elementi possono essere relazioni di **inclusione**, di **ordine** e di **ricorrenza**: per esempio potremmo dire che un elemento paragrafo è incluso in un elemento capitolo e può ricorrere molteplici volte, oppure che un elemento introduzione deve precedere un elemento capitolo, ecc.);
- ✓ alle foglie corrispondono gli elementi finali (generalmente i caratteri del testo).

Un esempio di albero gerarchico per un semplice documento è: nodo radice corrispondente al testo stesso; testo composto da un'eventuale introduzione, seguita da un certo numero di capitoli; a loro volta capitoli composti, ognuno, da un titolo, seguito da uno o più paragrafi; nodi terminali, o foglie, contenenti stringhe di caratteri costituenti il testo stesso (indicati con PCDATA) o altri elementi marcati, a seconda delle esigenze del *markup*.



Il concetto di classe

Fondandosi su di una codifica di tipo dichiarativo, SGML consente di **definire (o di dichiarare), in modo assolutamente personale ed autonomo, un insieme di marcatori (*tags*)** che adempiano al compito primario dello standard, che consentano cioè di fare il *markup* della struttura logica del documento.

Questo gruppo di marcatori, più precisamente, individua una **classe** di documenti testuali che presentano le medesime caratteristiche strutturali.

Con **classe intenderemo un'insieme di documenti che condividono determinate proprietà**: i testi poetici, i testi narrativi, i testi drammatici sono per esempio tre classi che possiedono una specifica struttura logica; una poesia sarà di conseguenza marcata secondo le proprietà della classe testi poetici.

Elementi, attributi e valori

Ogni porzione testuale può essere dunque individuata e descritta tramite ricorso ad un nome convenzionale, scelto dal codificatore, che prende il nome di **elemento** (p.e. ‘p’ per paragrafo, ‘n’ per nota ecc.) racchiuso tra due **delimitatori**; la porzione strutturale viene quindi rappresentata tramite un marcatore o *tag* di apertura ed uno di chiusura.

Per i *tags*, la sintassi di SGML ci suggerisce una forma grafica simbolica che possiamo decidere di rispettare o meno: parentesi ad angolo, o delimitatori, racchiudenti il nome dell’elemento, cioè l’identificatore dell’elemento, come *tag* d’apertura (`<elemento>`); stesse parentesi ma con l’identificatore preceduto dal simbolo dello slash / per il *tag* di chiusura: `</elemento>`).

È possibile anche associare agli elementi degli **attributi**, che possiedono un determinato **valore**.

Struttura sintattica di un documento conforme ad SGML

modello della classe

```
<poesia>
  <titolo>Titolo della poesia</titolo>
  <strofa numero="1" tipo="terzina">
     <verso numero="1">Questo è il primo verso</verso>
     <verso numero="2">Questo è il secondo verso</verso>
     <verso numero="3">Questo è il terzo verso</verso>
   </strofa>
</poesia>
```

Il giorno dei morti

Io vedo (come è questo giorno, oscuro!)
vedo nel cuore, vedo un camposanto
con un fosco cipresso alto sul muro.

implementazione del modello

```
<poesia>
  <titolo>Il giorno dei morti</titolo>
  <strofa numero="1">
     <verso numero="1">Io vedo (come è questo giorno, oscuro!)</verso>
     <verso numero="2">vedo nel cuore, vedo un camposanto</verso>
     <verso numero="3">con un fosco cipresso alto sul muro.
</verso>
   </strofa>
</poesia>
```

Funzioni di SGML

Più esattamente diremo che SGML è dotato di una sintassi astratta che spiega come operare il *markup* di un documento testuale, fornendo **regole che istruiscono l'utente su come aggiungere i marcatori** (cioè si dovranno porre una *tag* iniziale e una *tag* finale ad ogni porzione testuale; ogni *tag* sarà costituita da un nome convenzionale, che identifica l'elemento, racchiuso tra due delimitatori), **senza però fornire norme specifiche sui nomi per elementi ed attributi**.

SGML non fornisce dunque nessuna indicazione circa il vocabolario per gli elementi, né speciali caratteri in qualità di delimitatori (non obbliga cioè l'uso delle parentesi angolate, suggerisce solo questo tipo di notazione simbolica).

SGML come metalinguaggio

Per questo è lecito affermare che SGML più che un linguaggio è un **metalinguaggio** che fornisce esclusivamente le **regole sintattiche** necessarie all'edificazione di altri linguaggi di *markup* di testi.

Dal momento che SGML ragiona per classi di documenti diremo che ciascuno di tali linguaggi derivati si configura come un peculiare **modello del testo** o, più esattamente, di un **insieme di testi aventi caratteristiche logico-strutturali analoghe**.

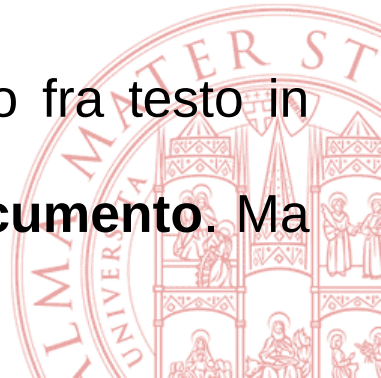
Il concetto di DTD

Lo standard non fornisce dunque alcuna prescrizione relativamente alla tipologia, alla quantità o al nome dei marcatori, ma si occupa esclusivamente di **fornire precise regole sintattiche**, necessarie a definire un insieme di marcatori.

Il valore di tali marcatori va specificato in un **vocabolario di marcatura**, che è detto "definizione del tipo di documento", o DTD (*Document Type Definition*). a quale classe appartiene, che tipo di struttura ha… ecc.

La DTD è quindi la **grammatica del metalinguaggio** o anche il linguaggio impiegato per la rappresentazione di determinati parametri logico-strutturali di gruppi di documenti aventi le medesime caratteristiche.

Si pensi alla tradizionale suddivisione per il testo letterario fra testo in prosa, testo in versi, testo drammatico, testo parlato, ecc.
Ciascuno di questo macro-raggruppamenti è un **tipo di documento**. Ma anche il testo letterario in sé può essere considerato tale.

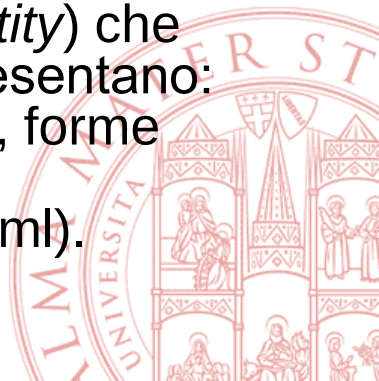


DTD e Schema di codifica

La DTD si chiama anche lo **schema di codifica** cui un testo fa riferimento – o anche il **modello** – che ha il compito di definire da un lato i diversi aspetti della fonte che possono essere oggetto di intervento interpretativo e, dall'altro, specificare il vocabolario associato a ciascuno degli aspetti.

Funzioni della DTD

- i marcatori per gli **elementi** (*elements*) che identificano la serie delle proprietà di un testo o di una certa classe di documenti; per ciascuno di tali elementi è definito un nome convenzionale che qualifica la proprietà strutturale della porzione della fonte che andrà ad identificare: p.e. ‘t’ per testo, ‘intro’ per introduzione, ‘cap’ per capitolo, ‘p’ per paragrafo, ‘n’ per nota, ecc.;
- la descrizione del **contenuto di ogni elemento** (*content model*), quindi quali altri elementi possono apparire, con quale ordine e con quale frequenza (una sezione sarà suddivisa in paragrafi, ogni un paragrafo potrà contenere parole e le parole potranno contenere lettere);
- i marcatori per gli **attributi** (*attributes*) assegnabili ad un qualsivoglia elemento (un titolo potrà essere caratterizzato dalla sua tipologia di livello, un capitolo dal suo numero progressivo, ecc.);
- i simboli (stringhe di testo ascii/Unicode) per le **entità** (*entity*) che possono occorrere all’interno del documento e che rappresentano: caratteri non esistenti nel *code set* impiegato alla codifica, forme contratte che vanno estese in fase di *layout* (per esempio un’abbreviazione), oggetti esterni (altri file sgml o non- sgml).



HTML come DTD

L'HTML (*HyperText Markup Language*) è un formato non proprietario basato sull'SGML, più precisamente è una **DTD SGML** che nasce quindi nel rispetto delle specifiche della sintassi dello standard e che prescrive un vocabolario legato a quella **classe** di documenti che sono gli **ipertesti.**

Nato agli inizi degli anni '90 e ideato da Tim Berner Lee, il padre fondatore del WWW. Specifica dal sito del consorzio W3C: http://www.w3.org/TR/#tr_HTML

Limiti di HTML

- **non modificabile**, quindi chiuso: l'autore può scegliere entro un numero predefinito di elementi, capaci di descrivere solo taluni fenomeni testuali e non ha la facoltà di esplicitarne di diversi.
- **scarsamente strutturato**, dotato di una sintassi poco potente, incapace di descrivere fenomeni complessi o informazioni altamente organizzate.
- spiccata predilezione per **marcatori stilistici più che strutturali**, cioè una codifica improntata alla descrizione dell'aspetto fisico del documento piuttosto che alla struttura logica. La maggior parte degli elementi del vocabolario HTML, fino alla versione 4.01, si concentra sulle caratteristiche visuali del documento (colore e dimensione dei font, tipo di caratteri, allineamento delle sezioni, colori e immagini di sfondo, ecc.).
- Ora con HTML5 alcuni problemi sono stati superati…

HTML Markup language

Non si tratta solo di diverse versioni ma di un modo di intendere il Web che si è evoluto ed è cambiato nel corso del tempo. Con l'evoluzione del Web sono cambiate le funzioni e le caratteristiche del linguaggio attraverso cui il Web si esprime.

Version	**Year**
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

HTML 4.01

```
<html>
<head>
<title>SCIENTIA RERUM</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="keywords" content="convegno, bologna, classici, scienze" />
    <meta name="description" content="scientia rerum la scienza di fronte ai classici convegno internazionale bologna" />
    <meta name="author" content="Francesca Tomasi" />
<link rel="stylesheet" href="stile.css" type="text/css">
</head>

<body>
<strong></strong>
<table width="80%" border="0">
  <tr>
    <td width="33%" height="154" rowspan="2" valign="top" bgcolor="#ced3e6"> <div align="center">
      </div>
      <div align="center">
        <p><font size="7" face="Georgia, Times New Roman, Times, serif"><img src="logoUNIBO.gif" width="182" height="48"></font><font color="#660033" size="1" face="Georgia, Times New Roman, Times, serif"><strong><br>
          </strong> <font color="#000000"><strong><br>
          <br>
          Centro Studi La permanenza del Classico</strong><br>
          Dipartimento di Filologia Classica e Medioevale Universit&agrave; degli
          Studi di Bologna <br>
          Via Zamboni, 32 - I - 40126 Bologna <br>
          Tel. 0512098539 - Fax 051228172<br>
          <br>
          <a href="http://www.classics.unibo.it/Permanenza">www.classics.unibo.it/Permanenza</a><br>
          <a href="mailto:permanenza@classics.unibo.it%20">permanenza@classics.unibo.it
          </a></font></font></p>
        </div></td>
    <td width="1%" height="154" rowspan="2"> <p> </p></td>
    <td width="66%" height="59" bgcolor="#f6f4f4">
      <div align="center">
        <p align="right"><em><font size="2" face="Geneva, Arial, Helvetica, sans-serif">Quid
          est bonum? Scientia rerum </font></em><font size="2" face="Geneva, Arial, Helvetica, sans-serif">(Seneca)</font></p>
      </div></td>
  </tr>
  <tr>
    <td height="157" bgcolor="#f6f4f4">
      <div align="center">
        <p><strong><font size="4" face="Georgia, Times New Roman, Times, serif">SCIENTIA
          RERUM</font><font size="7" face="Georgia, Times New Roman, Times, serif"><br>
          </font><font face="Georgia, Times New Roman, Times, serif">La <font color="#990000">scienza</font>
          di fronte ai <em>Classici</em></font></strong></p>
        <p><strong><font face="Georgia, Times New Roman, Times, serif"><strong>Bologna,
          29-30 settembre - 1 ottobre 2005<br>
          <font size="2">Aula Magna di Santa Lucia, via Castiglione 36 - BOLOGNA</font></strong></font></strong></p>
      </div></td>
  </tr>
  <tr>
    <td height="266" valign="top" bgcolor="#ced3e6">
      <div align="center">
        <p align="left"> </p>
        <p align="left"><font color="#000000"><b><img src="../../images/freccia.gif" width="8" height="9"><font size="1" face="Georgia, Times New Roman, Times, serif"><a href="programma.htm">Programma</a></font></b></font></p>
      </div>
      <p align="left"><font color="#000000" size="1" face="Georgia, Times New Roman, Times, serif"><strong><b><img src="../../images/freccia.gif" width="8" height="9"></b><a href="curricula.htm">Curricula
        dei relatori</a></strong></font> </p>
      <p align="left"><font color="#000000" size="1" face="Georgia, Times New Roman, Times, serif"><strong><b><img src="../../images/freccia.gif" width="8" height="9"></b><a href="handout.htm">Handout</a></strong></font>
      </p>
      <p align="left"><font color="#000000"><strong><img src="../../images/freccia.gif" width="8" height="9"><font size="1" face="Georgia, Times New Roman, Times, serif"><a href="comitato.htm">Comitato
        scientifico </a></font></strong></font></p>
      <p align="left"><font color="#000000"><strong><font color="#000000"><strong><img src="../../images/freccia.gif" width="8" height="9"><font size="1" face="Georgia, Times New Roman, Times, serif"><a href="patrocini sponsors.htm"
        e sponsors</a></font></strong></font><font size="1" face="Georgia, Times New Roman, Times, serif"><br>
        </font></strong></font></p></td>
    <td width="1%" height="266">
      <div align="center">
        <p> </p>
      </div>
      <p> </p></td>
    <td bgcolor="#f6f4f4"><div align="center">
```

XHTML overview

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>Documento di prova</title>
        <link rel="stylesheet" href="stile.css"/>
    </head>
    <body>
        <div id="container">
            <div id="header">
               <h1>Testata del sito</h1>
            </div>
        <div id="menu-principale">
               <ul class="toc">
                   <li><a href="index.htm">Home</a></li>
                   <li><a id="activelink">Pagina attiva</a></li>
                   <li><a href="pagina1.htm">Pagina 1</a></li>
               </ul>
        </div>
             <div id="contenuto">
                 <hr/>
                <p>E qui metto il testo che voglio compaia come corpo centrale</p>
                <p>E magari voglio un'immagine <img src="immagine.png"/></p>
                <p>Altro paragrafo con <strong>una parola enfatizzata</strong></p>
            </div>
             <div id="piede-pagina">
                <p>&copy; Data, <span class="persona">Nome cognome</span></p>
                <p>E ancora altre informazioni: contatti, credits, validazione</p>
                <p><a href="">Link1</a> | <a href="">Link2</a> | <a href="">Link3</a></p>
            </div>
        </div>
    </body>
</html>
```

con il valore di class riesco a definire qualcosa in più

HTML5. Struttura vs layout

Edit This Code: | See Result » | Result:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5</title>
<meta charset="utf-8">

<style>
body {font-family: Verdana, sans-serif; font-size:0.8em;}
header, nav, section, article, footer {border:1px solid grey; margin:5px; padding:8px;}
nav ul {margin:0; padding:0;}
nav ul li {display:inline; margin:5px;}
</style>
</head>

<body>

<header>
<h1>HTML5 Skeleton</h1>
</header>

<nav>
<ul>
  <li><a href="html5_semantic_elements.asp">HTML5 Semantic</a></li>
  <li><a href="html5_geolocation.asp">HTML5 Geolocation</a></li>
  <li><a href="html5_canvas.asp">HTML5 Graphics</a></li>
</ul>
</nav>

<section>
<h2>Famous Cities</h2>

<article>
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</article>

</section>

<footer>
<p>&copy; 2014 W3Schools. All rights reserved.</p>
</footer>

</body>
</html>
```

HTML5 Skeleton

HTML5 Semantic HTML5 Geolocation HTML5 Graphics

Famous Cities

London

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Paris

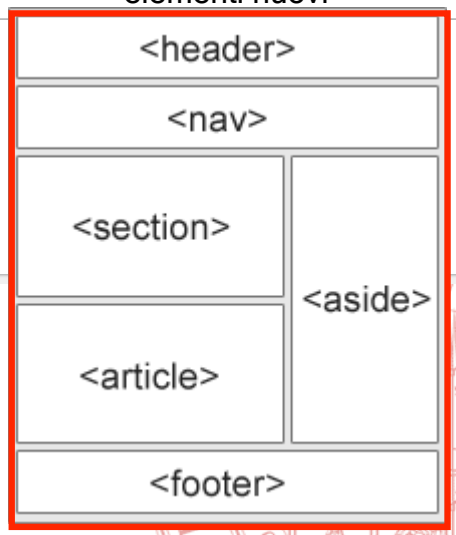
Paris is the capital and most populous city of France.

Tokyo

Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.

© 2014 W3Schools. All rights reserved.

elementi nuovi



- HTML5 verso la semantica della rappresentazione digitale.
- Fogli di stile CSS (*Cascading Style Sheets*) per la gestione dell'aspetto del documento.
- Necessità di separare struttura logica e contenuto dall'aspetto o layout del documento.

Parte pratica

E ora, prima di passare a XML, ci fermiamo e ci dedichiamo alla pratica:

Verifichiamo di aver installato SublimeText (https://www.sublimetext.com/)

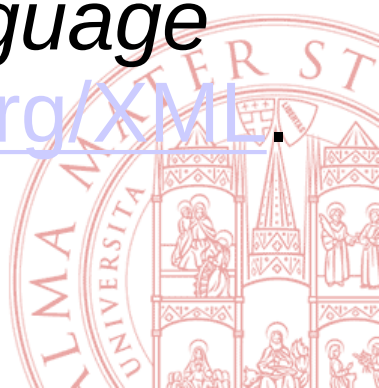
Per HTML5 useremo:
https://www.w3schools.com/html/
Usa anche: riassunto elementi HTML
(file 3-bis-elementi_attributi_HTML.pdf)

Per CSS useremo:
https://www.w3schools.com/css/default.asp

Nascita XML

Il W3C ha optato per la realizzazione di una versione semplificata dello standard ISO SGML che ha portato allo sviluppo dell'XML.

Il progetto ha avuto inizio alla fine del 1996, nell'ambito della *SGML Activity* del W3C e nel Febbraio del 1998 le specifiche sono divenute una raccomandazione ufficiale, con il nome *Extensible Markup Language* (XML) versione 1.0 http://www.w3.org/XML.



XML e *markup* strutturale

XML è dunque un sottoinsieme di SGML semplificato ed ottimizzato specificamente per applicazioni in ambiente Word Wide Web.

Si tratta dunque, come il suo predecessore, di un metalinguaggio, che permette di specificare molteplici **classi di linguaggi di marcatura**, e non quindi una semplice applicazione SGML come HTML.

La grande novità che caratterizza XML è la **descrizione logica delle informazioni testuali in un formato leggibile e comprensibile dall'utente, prescindendo dalle indicazioni relative a come i dati devono essere visualizzati**; solo in un secondo momento i dati marcati in XML possono ricevere istruzioni circa le modalità di visualizzazione e di formattazione.

XSL

Pur supportando XML i fogli di stile a cascata (CSS) esiste già una specifica per i fogli di stile, pensata per XML: l'*Extensible Stylesheet Language* XSL http://www.w3.org/Style/XSL/.

Usando l'XSL si assicura che i documenti XML risultino formattati nello stesso modo, indipendentemente dall'applicazione che si utilizza o dalla piattaforma di visualizzazione.

Il foglio di stile, oltre ad assegnare un certo *layout* al documento, consente di amministrare viste diverse sullo stesso file.

XML, struttura logica, albero gerarchico

Un *markup* XML esprime quindi il valore della stringa di caratteri cui il *tag* è associato a prescindere dalle modalità di resa grafica.

Diremo che XML focalizza la codifica sulla **struttura** e quindi sul valore dei blocchi logici, documentando l'ordinamento **gerarchico** che sovrintende all'organizzazione degli elementi della fonte.

DTD e XML *Schema*

A seconda delle esigenze del *markup* e della classe di appartenenza del documento, è possibile specificare in modo personale la serie dei marcatori utili alla descrizione del documento.

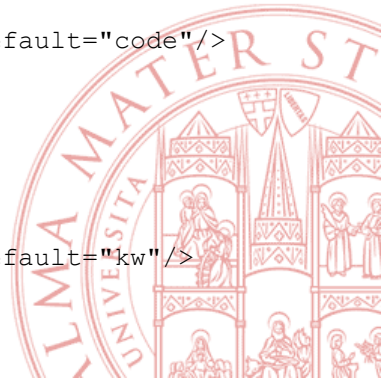
I *tags* utili possono essere quindi creati a piacimento e ciascun elemento essere poi specificato nella definizione del tipo di documento (DTD) o in un XML *Schema*, un *constraint language* che permette di descrivere la struttura dei documenti XML tramite la stessa sintassi XML (cf. http://www.w3.org/XML/Schema).

Si tratta del sistema elaborato dal W3C, al fine di consentire di superare i limiti della DTD, che, essendo scritta con una sintassi differente da quella dell'XML, non può essere trattata con gli stessi strumenti software per l'XML.

DTD vs XML Schema

```
<!ELEMENT eg (#PCDATA)*>
<!ATTLIST eg
  xmlns CDATA #FIXED 'http://www.tei-c.org/ns/1.0'
  %attributes.class.global;
  TEIform CDATA 'eg'>

<!ELEMENT gi (#PCDATA)*>
<!ATTLIST gi
  xmlns CDATA #FIXED 'http://www.tei-c.org/ns/1.0'
  %attributes.class.global;
  TEI (yes|no) 'yes'
  TEIform CDATA 'gi'>

<!ELEMENT code (#PCDATA)*>
<!ATTLIST code
  xmlns CDATA #FIXED 'http://www.tei-c.org/ns/1.0'
  %attributes.class.global;
  type CDATA #IMPLIED
  TEIform CDATA 'code'>

<!ELEMENT kw (#PCDATA)*>
<!ATTLIST kw
  xmlns CDATA #FIXED 'http://www.tei-c.org/ns/1.0'
  %attributes.class.global;
  TEIform CDATA 'kw'>

<!ELEMENT ident (#PCDATA)*>
<!ATTLIST ident
  xmlns CDATA #FIXED 'http://www.tei-c.org/ns/1.0'
  %attributes.class.global;
  type CDATA #IMPLIED
  TEIform CDATA 'ident'>
```

```
<xs:element name="eg">
    <xs:complexType mixed="true">
      <xs:attributeGroup
ref="ns1:attributes.class.global"/>
      <xs:attribute name="TEIform" default="eg"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="gi">
    <xs:complexType mixed="true">
      <xs:attributeGroup
ref="ns1:attributes.class.global"/>
      <xs:attribute name="TEI" default="yes">
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="yes"/>
            <xs:enumeration value="no"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="TEIform" default="gi"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="code">
    <xs:complexType mixed="true">
      <xs:attributeGroup
ref="ns1:attributes.class.global"/>
      <xs:attribute name="type"/>
      <xs:attribute name="TEIform" default="code"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="kw">
    <xs:complexType mixed="true">
      <xs:attributeGroup
ref="ns1:attributes.class.global"/>
      <xs:attribute name="TEIform" default="kw"/>
    </xs:complexType>
  </xs:element>
```

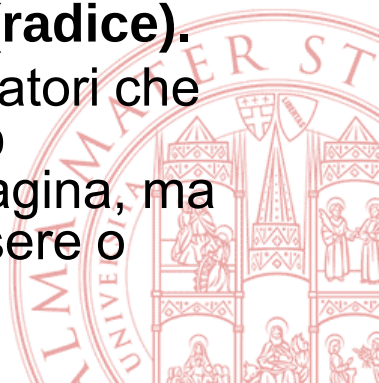


Documenti
VALIDI e BEN FORMATI

- Dunque anche un singolo documento XML valido può essere associato ad uno **schema** (ci riferiremo con questa parola sia alla DTD che all'XML *Schema*) che ne specifica la grammatica, nel senso di vocabolario degli elementi e relazioni gerarchiche fra gli stessi.
- Il documento XML che fa riferimento ad uno schema si dice '**valido'**.
- Tuttavia a differenza di SGML, XML consente la distribuzione anche di documenti privi di schemi, documenti *well-formed*, cioè '**ben formati**', che rispettano le regole XML per la formazione e la collocazione dei marcatori, pur senza richiedere il riferimento ad uno schema.

Caratteristiche sintattiche di XML

- ✓ XML richiede il rispetto di regole sintattiche rigide, che consentono però la leggibilità del file risultante su qualsiasi piattaforma.
- ✓ XML è *case sensitive*: <ELEMENTO></ELEMENTO> è corretto e lo è anche <elemento></elemento> ma non lo è <elemento></ELEMENTO>. Il *case* (minuscolo o maiuscolo) utilizzato per i nomi degli elementi deve essere quindi il medesimo in apertura e in chiusura.
- ✓ Il valore dell'attributo va sempre posto fra virgolette: <immagine riferimento="img.jpg"/>
- ✓ Si rende necessario un corretto annidamento dei marcatori: NON <a> MA <a>. Questo vincolo dell'XML, derivato dall'SGML, limita la rappresentazione di fenomeni testuali concorrenti e impedisce la gestione di due strutture che si intersecano, cioè delle gerarchie sovrapposte.
- ✓ Deve esistere un tag che contiene tutti gli altri, il tag ***root* (radice).**
- ✓ Possono essere utilizzati elementi 'vuoti', vale a dire marcatori che non racchiudono blocchi o porzioni di testo ma forniscono un'indicazione come la fine di una riga, il cambio di una pagina, ma anche l'inserimento di un'immagine ecc. La forma può essere o <tagvuoto/> oppure <tagvuoto></tagvuoto>.



Per riassumere

XML è dunque un metalinguaggio che definisce le **norme sintattiche** per effettuare il *markup* di un documento, ma non fornisce alcun criterio per la scelta e la denominazione degli elementi utili a descrivere i fenomeni testuali.

E' possibile utilizzare uno **schema**, che documenta il vocabolario di elementi e le relazioni fra gli stessi; ma è anche lecito creare **marcatori di invenzione**, quando uno schema esistente non sia sufficiente a rappresentare le caratteristiche della fonte che si intendono codificare.

Una volta definite tali caratteristiche si possono individuare e definire dei **nomi convenzionali per gli elementi** che esplicitano il valore delle porzioni di testo cui andranno riferiti.

Se la prerogativa dei linguaggi di *markup* dichiarativo è la codifica della struttura in realtà moltissimi **fenomeni possono essere oggetto di descrizione**: essi configurano il **modello** della fonte, risultato della selezione delle informazioni veicolate dal testo che si intendono marcare, cioè descrivere.

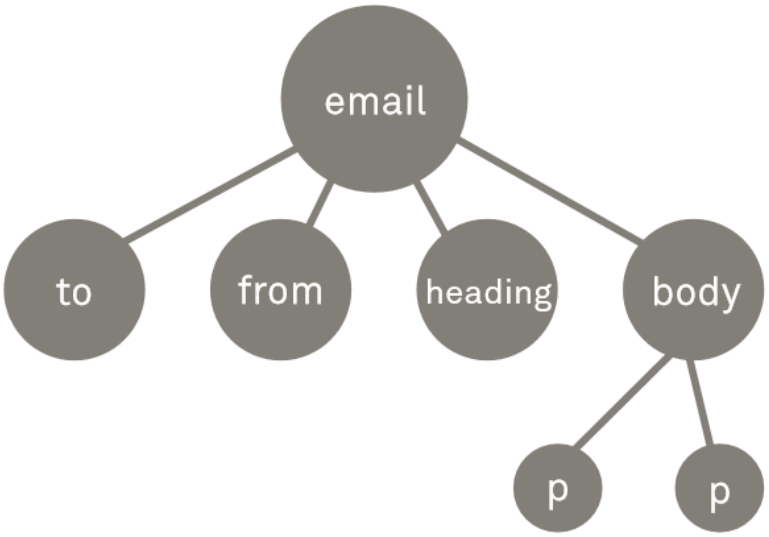
Potranno quindi esistere molteplici codifiche XML diverse, ciascuna rappresentativa di uno specifico modello e, in potenza, nessun modello potrà esaurire gli aspetti passibili di intervento interpretativo.

Passiamo alla pratica

struttura di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>
<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don’t forget me this weekend!</p>
        <p>We have to do homeworks.</p>
    </body>
</email>
```

ogni documento XML (.xml) è caratterizzato da una **struttura gerarchica**, ad albero, composta da **nodi** (o **elementi**)



struttura di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don’t forget me this weekend!</p>
        <p>We have to do homeworks.</p>
    </body>
</email>
```

ogni documento XML (.xml) è caratterizzato da una **struttura gerarchica**, ad albero, composta da **nodi** (o **elementi**)

- un nodo **radice**

racchiude tutti gli altri nodi del documento ed in genere il suo nome è rappresentativo dell’oggetto che descrive

struttura di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don’t forget me this weekend!</p>
        <p>We have to do homeworks.</p>
    </body>
</email>
```

ogni documento XML (.xml) è caratterizzato da una **struttura gerarchica**, ad albero, composta da **nodi** (o **elementi**)

- un nodo **radice**

- tutti gli altri nodi sono **discendenti** del nodo radice

struttura di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don’t forget me this weekend!</p>
        <p>We have to do homeworks.</p>
    </body>
</email>
```

ogni documento XML (.xml) è caratterizzato da una **struttura gerarchica**, ad albero, composta da **nodi** (o **elementi**)

- un nodo **radice**

- tutti gli altri nodi sono **discendenti** del nodo radice

- ogni nodo può:
1. contenere uno o più nodi

struttura di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don’t forget me this weekend!</p>
        <p>We have to do homeworks.</p>
    </body>
</email>
```

ogni documento XML (.xml) è caratterizzato da una **struttura gerarchica**, ad albero, composta da **nodi** (o **elementi**)

- un nodo **radice**

- tutti gli altri nodi sono **discendenti** del nodo radice

- ogni nodo può:
1. contenere uno o più nodi
2. contenere stringhe di caratteri

(oppure 3. entrambi)

struttura di un documento XML

```
<?xml version="1.0" encoding="UTF-8"?>

<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don't forget me this weekend!</p>
        <p>We have to do homeworks.</p>
    </body>
</email>
```

ogni documento XML (.xml) ha un **prologo**, contenente:

- la versione del linguaggio utilizzato (1.0, la versione corrente);

- la codifica dei caratteri utilizzata (UTF-8, Unicode Transformation Format, 8 bit)

<? ?> è il tag che racchiude le (uniche) istruzioni di un file XML.

sintassi di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don’t forget me this weekend!</p>
        <p>We have to do homeworks.</p>
        <sign/>
    </body>
</email>
```

ogni elemento (o tag) contenente del testo, o altri elementi, deve essere **aperto e chiuso**, con una precisa notazione, andando a racchiudere il suo contenuto

sintassi di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don’t forget me this weekend!</p>
        <p>We have to do homeworks.</p>
        <sign/>
    </body>
</email>
```

ogni elemento (o tag) contenente del testo, o altri elementi, deve essere **aperto e chiuso**, con una precisa notazione, andando a racchiudere il suo contenuto

il testo contenuto in un elemento è detto **valore dell’elemento**

sintassi di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don’t forget me this weekend!</p>
        <p>We have to do homeworks.</p>
        <sign/>
    </body>
</email>
```

ogni elemento (o tag) contenente del testo, o altri elementi, deve essere **aperto e chiuso**, con una precisa notazione, andando a racchiudere il suo contenuto

il testo contenuto in un elemento è detto **valore dell’elemento**

un elemento che non contiene alcun valore, può essere scritto in **forma abbreviata**

sintassi di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email>
    <to>Mark</to>
    <from>Veronika</from>
    <heading>Reminder</heading>
    <body>
        <p>Don’t forget me this weekend!</p>
        <p>We have to do homeworks.</p>
        <sign/>
    </body>
</email>
```

ogni elemento contenuto in un altro elemento è detto **annidato**

l’annidamento è rigorosamente **gerarchico**: un elemento può essere racchiuso in un altro elemento solo nella sua totalità

NON è corretto ad esempio:

```
<heading>Reminder
<body>Don’t forget</heading></body> me...
```

sintassi di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email to=“Mark” from=“Veronika” sign=“no”>
    <heading>Reminder</heading>
    <body>
        <p n=’1’>Don’t forget me this weekend!</p>
        <p n=’2’>We have to do homeworks.</p>
    </body>
</email>
```

N.B. non tutte le informazioni devono essere espresse come elementi

gli **attributi** sono una forma alternativa, a volte più efficiente, per descrivere gli stessi contenuti

si parla di attributo (@to)
e **valore dell’attributo** (‘Mark’)

un attributo si riferisce all’elemento in cui è contenuto ed è riportato solo nel **tag di apertura**

sintassi di un documento XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email to=“Mark” from=“Veronika” sign=“no”>
    <heading>Reminder</heading>
    <body>
        <p n=’1’>Don’t forget me this weekend!</p>
        <p n=’2’>We have to do homeworks.</p>
    </body>
</email>
```

XML è **case sensitive**
<email> non è uguale a <EMAIL>

l’indentazione è una buona prassi per la leggibilità del documento

N.B. la sintassi (non la struttura) di un file XML è molto simile a quella di un file HTML, dove abbiamo:

- elemento radice fisso: <html>
- elementi annidati
- tag aperti e chiusi
- attributi di elementi

schemi XML

```
<?xml version=”1.0” encoding=”UTF-8”?>

<email xmlns=”http://www.myscheme.org/1.0”
to=“Mark” from=“Veronika” sign=“no”>
    <heading>Reminder</heading>
    <body>
        <p n=’1’>Don’t forget me this weekend!</p>
        <p n=’2’>We have to do homeworks.</p>
    </body>
</email>
```

gli schemi sono dei documenti reperibili sul web e identificati da una **URL**

quando utilizziamo uno (o più) schemi, specifichiamo nell’elemento radice l’attributo @xmlns la URL, ovvero il suo **namespace**

Al lavoro… Per produrre un file XML ben formato

- ✓ Andiamo su OTA - https://ota.ox.ac.uk/
- ✓ Accediamo al catalogo
- ✓ Il catalogo stesso potrebbe essere rappresentato in XML
- ✓ Apriamo *The Winters Tale* di Shakespeare (http://ota.ox.ac.uk/desc/5729)
- ✓ Copiamo e incolliamo in Sublime Text…
- ✓ Aggiungiamo il prologo <?xml version="1.0" encoding="UTF-8"?>
- ✓ Mettiamo la root <collezione> e il primo item <oggetto n="1">
- ✓ Salviamo il file con un nome e l'estensione .xml
- ✓ Apriamo con il browser
- ✓ Errore su & … sostituiamo con &
- ✓ Modifichiamo e costruiamo il nostro file .xml in Sublime….

La macro struttura ovvero il modello della descrizione

Prologo

Radice

- Oggetto
  - Titolo
  - Autore
  - Accesso
    - licenza
    - formati
    - analisi
  - Lingua
  - Parole-chiave
    - soggetto
  - Descrizione
    - [riferimenti-bibliografici, note, persone, titoli, etc…]
  - Identificativo

Il file .xml

```
<?xml version="1.0" encoding="UTF-8"?>
<collezione>
    <oggetto n="1">
        <titolo>The Winters Tale.</titolo>
        <autore permalink="https://viaf.org/viaf/96994048">Shakespeare, William, 1564-1616</autore>
        <accesso>
            <licenza>Distributed by the University of Oxford under a
                      <URL indirizzo="https://creativecommons.org/licenses/by-sa/3.0/">Creative Commons Attribution-ShareAlike
                       3.0 Unported License</URL>
            </licenza>
            <formati>Download: XML; HTML; ePub; mobi (Kindle); plain text</formati>
            <analisi>Analysis: Explore this text with Voyant Tools (this link takes you to the
                     <URL>voyant-tools.org</URL> website - find out more here)
            </analisi>
        </accesso>
        <lingua>English</lingua>
        <parole-chiave>
            <soggetto vocabolario="LC" xml:id="sogg1">Plays -- England -- 16th century</soggetto>
            <soggetto vocabolario="LC" xml:id="sogg2">Plays -- England -- 17th century</soggetto>
            <soggetto vocabolario="LC" xml:id="sogg3">Comedies -- England -- 16th century</soggetto>
            <soggetto vocabolario="LC" xml:id="sogg4">Comedies -- England -- 17th century</soggetto>
            <soggetto vocabolario="LC" xml:id="sogg5">Tragedies -- England -- 16th century</soggetto>
            <soggetto vocabolario="LC" xml:id="sogg6">Tragedies -- England -- 17th century</soggetto>
      </parole-chiave>
      <descrizione-fonte>
      Revised version of http://ota.ox.ac.uk/id/0119. The texts were originally prepared by Trevor Howard-Hill for use in his
single volume concordances to Shakespeare (OUP, 1969f). They have since been reformatted to modern standards and carefully
proofread by staff of Oxford University Press' Shakespeare Department for use in the new "Old Spelling" Oxford Shakespeare, under
the general editorship of Dr Stanley Wells: The complete works / William Shakespeare; general editors, Stanley Wells and Gary
Taylor ; editors Stanley Wells ... [et al.] ; with introductions by Stanley Wells. -- Oxford : Clarendon Press, 1986. -- (Oxford
Shakespeare). -- ISBN 0-19-812926-2.
            <fonte>The Winters Tale. Shakespeare, William, 1564-1616 Lee, Sidney, Sir, 1859-1926 xxxv, 908 p. : facsims. ; 39 cm.
Clarendon Press Oxford: 1902</fonte>
            <nota n="1"></nota>
            <nota n="2"></nota>
            <nota n="3"></nota>
        </descrizione-fonte>
        <identificativo>http://purl.ox.ac.uk/ota/5729</identificativo>
    </oggetto>
</collezione>
```

Bibliografia storica sui linguaggi di markup

- BARNARD D.T. *et al.* (1988), *SGML-Based Markup for Literary Texts: Two Problems and Some Solutions*, "Computer and the Humanities", XXII, pp. 265-276.
- CIOTTI F. (1994), *Il testo elettronico. Memorizzazione, codifica ed edizione* in *Macchine per leggere. Tradizioni e nuove tecnologie per comprendere i testi*, Atti del convegno, Fondazione Ezio Franceschini, Centro italiano di studi sull'alto medioevo, Spoleto, pp. 213-230.
- COOMBS J.H., RENEAR A.H., DEROSE S.J. (1987), *Markup system and the future of scholarly text processing*, "Communications of the ACM 30 (CACM)", vol. XXX, n. XI, pp. 933-47, URL=<http://www.oasis-open.org/cover/coombs.html>.
- DE ROSE S.J. *et al.* (1990), *What is Text, Really?*, in "Journal of Computing in Higher Education", 1/2, pp. 3-26.
- DE ROSE S.J. (2004), *Markup Overlap: A Review and a Horse*, in "Proceedings of Conference on Extreme Markup Languages".
- FIORMONTE D. (2003), *Scrittura e filologia nell'era digitale*, Bollati Boringhieri, Torino.
- GOLDFARB C.F. (1990), *The SGML Handbook*, Oxford, Oxford University Press.
- ORLANDI T. (1990), *Informatica umanistica*, Nuova Italia Scientifica, Roma.
- PIERAZZO E. (2005), *La codifica dei testi*, Carocci, Roma.
- RENEAR A.H., DUBIN D., SPERBERG-MCQUEEN C.M., AND HUITFELDT C. (2003), *Towards a semantics for XML markup*, in R. Furuta, J. I. Maletic, E. Munson (ed. by), *Proceedings of the 2002 ACM symposium on Document engineering*, New York, ACM Press, pp. 119-126.