ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

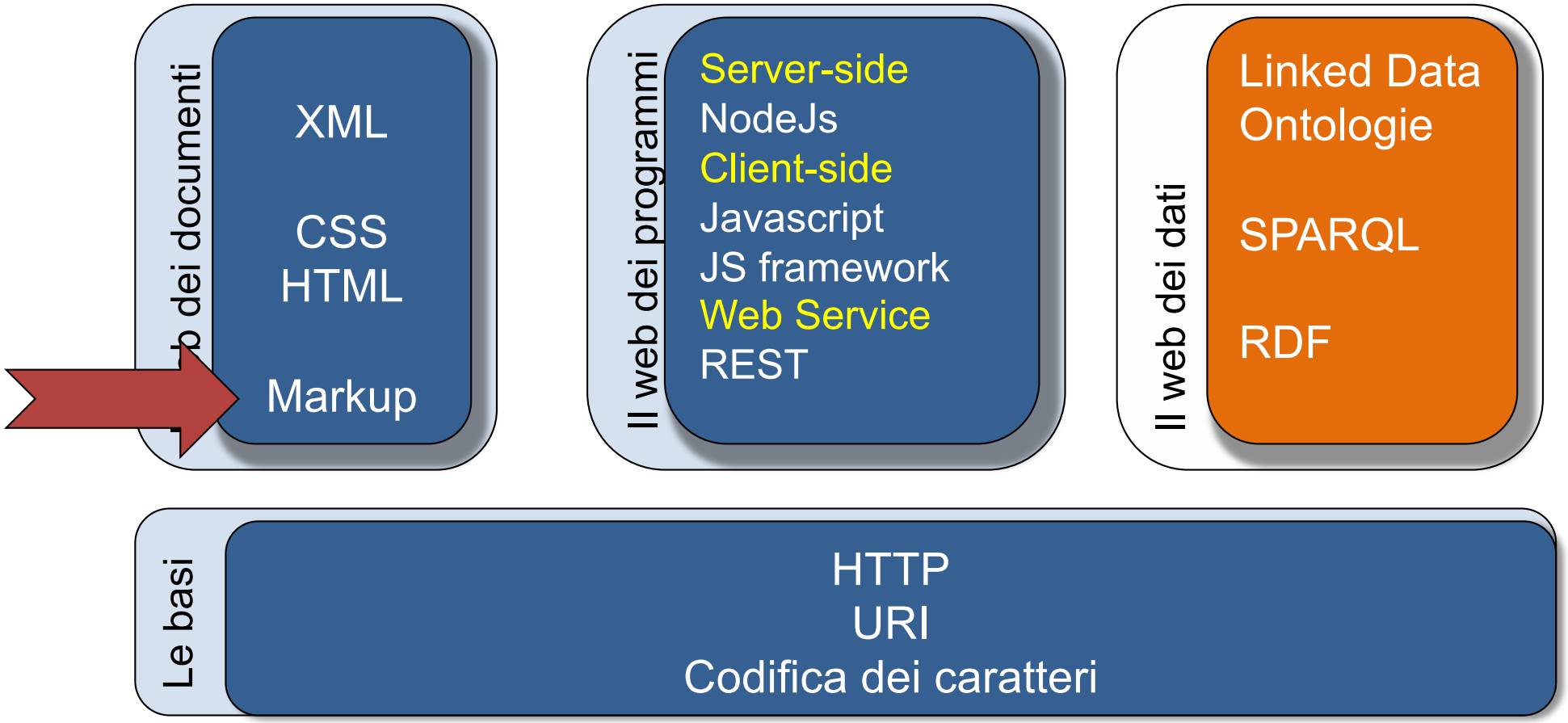
Markup

Fabio Vitali

CdS in Informatica

CdS in Informatica per il Management

Argomenti delle lezioni



«Quando io uso una parola», disse Humpty Dumpty in tono piuttosto sprezzante, «significa quello che io scelgo che significhi - né più, né meno.»

Lewis Carroll

"Attraverso lo specchio"

Introduzione

Qui esaminiamo:

- Cos’è il markup
- Una storia del markup su computer
- Alcuni linguaggi di markup
- SGML e XML

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Cos'è il markup

Cos’è il markup? (1)

Definiamo markup ogni mezzo per rendere esplicita una particolare interpretazione di un testo.

Per esempio, tutte quelle aggiunte al testo scritto che permettono di renderlo più fruibile.

Oltre a rendere il testo più leggibile, il markup permette anche di specificare ulteriori usi del testo.

Con il markup per sistemi informatici (il nostro caso), specifichiamo le modalità esatte di utilizzo del testo nel sistema stesso.

Il markup non è soltanto un inevitabile e sgradevole risultato della informatizzazione dell’arte tipografica. Non è qualcosa che sta con noi a causa dell’informatica.

Cos’è il markup? (2)

Quando un autore scrive, da millenni a questa parte, specifica anche i delimitatori di parola (chiamati spazi), i delimitatori di frase (chiamati virgole) e i delimitatori di periodo (chiamati punti).

La numerazione delle pagine o l’uso dei margini per creare effetti sul contenuto sono noti da centinaia di anni.

Eppure questo a stretto rigore non fa parte del testo, ma del markup: nessuno dirà ad alta voce ‘virgola’ o ‘punto’ nel leggere un testo, ma creerà adeguati comportamenti paralinguistici (espressioni, toni, pause) per migliorare in chi ascolta la comprensione del testo.

La piramide dei linguaggi di markup

Modi del markup: proprietario vs. pubblico

- Un formato proprietario è stato creato da una specifica azienda con uno specifico scopo commerciale. L'azienda ne detiene i diritti, e dunque è in grado di modificarlo, aggiornarlo o rivoluzionarlo in qualunque momento e per qualunque motivo.
- Un formato pubblico è stato creato da un gruppo di interesse (individui, aziende, enti non commerciali, ecc.) come modello di armonizzazione tra le esigenze di ciascun partecipante.
- Il gruppo tipicamente pubblica le specifiche del formato, permettendo a chiunque di realizzare strumenti software per quel formato. A volte questo si concretizza in uno standard ufficiale, avente valore normativo.

Modi del markup: binario vs. leggibile

- Un formato binario è la memorizzazione esatta delle strutture in memoria dell'applicazione, che niente hanno a che vedere con le esigenze di comprensione di esseri umani. Il testo non è visibile o è visibile per caso.
- Un formato leggibile invece è fatto per essere, in casi speciali, letto anche da esseri umani, che possono intervenire per operazioni di emergenza.
- L'applicazione deve trasformare quanto legge in una struttura interna utile per le operazioni di modifica o presentazione. Questa fase si chiama parsing.

Modi del markup: interno vs. esterno

- Il markup interno inserisce istruzioni di presentazione all'interno del testo, in mezzo alle parole.
- Il markup esterno prevede due blocchi di informazioni: il contenuto e il markup, separati e collegati da indirezione.
- Il markup interno richiede sintassi particolari per distinguere il markup dal contenuto. Tipicamente si adottano segnalatori particolari che cambiano il tipo di interpretazione del documento. La presenza del carattere segnalatore nel testo richiede l'adozione di tecniche di escaping.
- Il markup esterno richiede un meccanismo di indirezione, basato su indirizzi, offset o identificatori, per associare con correttezza il markup al contenuto.

Modi del markup: procedurale vs. descrittivo

Il markup assolve a diversi ruoli a seconda del sistema di elaborazione, dell'applicazione, dello scopo a cui il documento è soggetto.

- Puntuazionale
- Presentazionale
- Procedurale
- Descrittivo
- Referenziale
- Metamarkup

Markup puntuazionale

- Il markup puntuazionale consiste nell'usare un insieme prefissato di segni per fornire informazioni perlopiù sintattiche sul testo.
- Le regole di punteggiatura sono sostanzialmente stabili, note agli autori, e frequenti nei documenti. Per questo gli autori tipicamente forniscono il loro markup puntuazionale autonomamente.
- Esistono tuttavia notevoli problemi nell'uso della punteggiatura:
  - Incertezze strutturali (virgola, punto e virgola o punto?),
  - Incertezze grafiche (virgolette aperte e chiuse o neutre?),
  - ambiguità procedurali (il punto viene usato sia per segnare la fine di una frase, che l'esistenza di un'abbreviazione, senza contare i tre puntini di sospensione).

Markup presentazionale

- Il markup presentazionale consiste nell'indicare effetti (grafici o altro) per rendere più chiara la presentazione del contenuto.
- Nel testo, possono essere cambi di paragrafo o di pagina, interlinea, pallini per liste, ecc.
- E' altresì markup presentazionale: cambiare pagina all'inizio di una nuova sezione, scrivere "Capitolo 3" in cima alla pagina, ecc.

Markup procedurale

- Il markup procedurale consiste nell'indicare con precisione ad un sistema automatico che effetto attivare e che procedura (serie di istruzioni) eseguire nella visualizzazione del contenuto.
- In definitiva, utilizzo le capacità del sistema di presentazione per avere con precisione l'effetto voluto.
- Esempio: *Wordstar Dot Commands*

```
.PL 66
.MT 6
.MB 9
.LH 12
```

Markup descrittivo

- Il markup descrittivo consiste nell'identificare strutturalmente il tipo di ogni elemento del contenuto.
- Invece di specificare effetti grafici come l'allineamento o l'interlinea, ne individuo il ruolo all'interno del documento, specificando che un elemento è un titolo, un paragrafo, o una citazione.

Markup referenziale

- Il markup referenziale consiste nel fare riferimento ad entità esterne al documento per fornire significato o effetto grafico ad elementi del documento. Per esempio, utilizzare una sigla nota che venga poi sostituita dalla parola intera durante la stampa
- Es.: l'autore scrive "CdL" e il sistema trasforma automaticamente l'input in "Corso di Laurea"

Metamarkup

- Il metamarkup consiste nel fornire regole di interpretazione del markup e permette di estendere o controllare il significato del markup.
- Ad esempio, la possibilità di definire macro per l'interpretazione e la visualizzazione del documento, o il processo di definizione degli elementi e delle procedure valide di un documento.

Un testo su carta

Tre Uomini in Barca

Jerome K. Jerome

1889

Capitolo primo

Tre invalidi - Le sofferenze di George e Harris - La vittima di centosette malattie inguaribili - [...]

Eravamo in quattro: George, William Samuel Harris, e io, Montmorency. Standocene seduti in camera mia, fumavamo e parlavamo di quanto fossimo malridotti… malridotti, dal punto di vista della salute, intendo, naturalmente.
Ci sentivamo tutti piuttosto giù di corda, ...

Un testo senza markup

Questo è il testo completamente senza markup, come poteva essere scritto su un papiro della biblioteca di Alessandria, nel II o III secolo a.C.

```
treuominiinbarcajeromekjerome1889capitolop
rimotreinvalidilesofferenzedigeorgeeharris
lavittimadicentosettemalattieinguaribilier
avamoinquattrogeorgewilliamsamuelharriseio
montmorencystandocenesedutiincameramiafuma
vamoeparlavamodiquantofossimomalridottimal
ridottidalpuntodivistadellasaluteintendona
turalmentecisentivamotuttipiuttostogiùdico
rda
```

Markup metabolizzato

Aggiungiamo markup puntuazionale e presentazionale: maiuscole/minuscole, punteggiatura, spazi e ritorni a capo sono essi stessi elementi di markup.

```
Tre Uomini in Barca
Jerome K. Jerome (1889)
Capitolo primo
Tre invalidi - Le sofferenze di George e Harris - La
vittima di centosette malattie inguaribili - […]
Eravamo in quattro: George, William Samuel Harris, e
io, Montmorency. Standocene seduti in camera mia,
fumavamo e parlavamo di quanto fossimo malridotti…
malridotti, dal punto di vista della salute, intendo,
naturalmente.
Ci sentivamo tutti piuttosto giù di corda, ...
```

Markup procedurale

Sono comandi, o istruzioni che il sistema di lettura (umano o elettronico) deve eseguire sul testo. Ad esempio, istruzioni su come andare a capo, come decidere i margini, ecc. Questo è RTF.

```
{\rtf1 \mac \ansicpg10000 \uc1  \pard \plain \s15 \qc \widctlpar
\adjustright \f4 \fs48 \cgrid {Tre Uomini in Barca \par } \pard \plain
\widctlpar \adjustright \f4 \cgrid { \line \par \par \par \par \par }
\pard \plain \s1 \qc \keepn \widctlpar \outlinelevel0 \adjustright \i
\f4 \fs36 \cgrid {Jerome K. Jerome \par } \pard \plain \qc \widctlpar
\adjustright \f4 \cgrid { \fs36 […] \par 1889 \line \par } \pard
\widctlpar \adjustright { \page } { \b \fs36 Capitolo primo} { \par
\par \par \line } { \i Tre invalidi - Le sofferenze di George e Harris
- La vittima di centosette malattie inguaribili - [ \u8230 \'c9] \par
} { \line } { \fs28 Eravamo in quattro: George, William Samuel Harris,
e io, Montmorency. Standocene seduti in camera mia, fumavamo e
parlavamo di quanto fossimo malridotti \u8230 \'c9 malridotti, dal
punto di vista della salute, intendo, naturalmente. \line Ci sentivamo
tutti piuttosto gi \u249 \'9d di corda, ... \par }}
```

Markup descrittivo

Sono informazioni (descrizioni) sugli elementi del documenti, che ne specificano il ruolo, la giustificazione, la relazione con gli altri elementi.

```
<ROMANZO>
  <TITOLO>Tre Uomini in Barca</TITOLO>
  <AUTORE>Jerome K. Jerome</AUTORE>
  <ANNO>1889</ANNO>
  <CAPITOLO>
    <TITOLO>Capitolo primo</TITOLO>
    <INDICE>
      <EL>Tre invalidi</EL>
      <EL>Le sofferenze di George e Harris </EL>
      <EL> La vittima di centosette malattie inguaribili </EL>
    </INDICE>
    <PARA>Eravamo in quattro: George, William Samuel Harris, e io,
Montmorency. Standocene seduti in camera mia, fumavamo e parlavamo di
quanto fossimo malridotti… malridotti, dal punto di vista della
salute, intendo, naturalmente. </PARA>
    <PARA>Ci sentivamo tutti piuttosto giù di corda, …</PARA>
  </CAPITOLO>…
</ROMANZO>
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

I linguaggi di markup

TROFF/NROFF (1)

Nato nel 1973, fa parte della distribuzione Unix standard. Sotto Linux si chiama Groff

E' ancora usato per documentazione tecnica, in particolare i manuali on-line di Unix

Esiste un formatter che è in grado di creare documenti stampabili sia su stampante (troff) che su schermo a carattere (nroff).

I comandi sono o esterni (compaiono su righe autonome precedute da un punto) o interni (introdotte dal carattere di escape "\")

Permette la definizione e l'uso di quattro tipi di carattere: roman, italic, bold e symbol.

Permette la creazione di macro complesse (il nome è al massimo di due caratteri, però)

TROFF /NROFF (2)

```
.\"  Questo è un esempio Troff.
.\"  margine sinistro 4 cm.
.\"  ampiezza del testo 8 cm.
.po 4c
.ll 8c
.\"  Inizia il documento.
.ft B
1. Introduzione a Troff

.ft P
Questo \(`e un esempio di documento scritt
elaborato con Troff.
In questo caso, si presume che verr\(`a ut
l'opzione \fB\-ms\fP).

.ft B
1.1 Paragrafi

.ft P
Il testo di un paragrafo termina quando ne
vuota.

Per la precisione, gli spazi verticali vengono rispettati, per cui le righe
vuote si traducono in spazi tra i paragrafi, anche quando queste sono pi\(`u di
una.
```

1. Introduzione a Troff

Questo è un esempio di documento scritto in modo tale da poter essere elaborato con Troff. In questo caso, si presume che verrà utilizzata lo stile "**s**" (con l'opzione **–ms**).

1.1 Paragrafi

Il testo di un paragrafo termina quando nel sorgente viene incontrata una riga vuota.

Per la precisione, gli spazi verticali vengono rispettati, per cui le righe vuote si traducono in spazi tra i paragrafi, anche quando queste sono più di una.

Questo è l'inizio di un nuovo paragrafo dopo tre righe vuote di separazione.

T_EX e $LaTeX$ (1)

Realizzato agli inizi degli anni 70 da Donald Knuth.

Linguaggio di programmazione completo, dotato di comandi per la formattazione di testi (circa 300 comandi fondamentali, detti primitive).

Di notevole complessità, è rivolta unicamente a programmatori e tipografi molto sofisticati.

Metafont è un sistema associato a TeX per la descrizione delle forme dei caratteri di un font attraverso formule matematiche.

TeX permette la generazione di macro che semplificano notevolmente la generazione di testi particolarmente sofisticati.

Esistono librerie di macro che permettono di scrivere documenti arbitrariamente complessi: matematica, chimica, musica, grafica vettoriale, grafica bitmap, ecc.

Nel 1985 Leslie Lamport sviluppò LaTeX, una raccolta di macro TeX per la generazione di una ventina circa di tipi di documento particolarmente comuni: articolo, libro, lettera, annuncio, ecc.

TEX e LATEX (2)

```
documentclass{article}

% Inizia il preambolo.

\setlength{\textwidth}{7cm}
\setlength{\textheight}{7cm}

% Fine del preambolo.

\begin{document}

% Inizia il documento vero e proprio.

\section{Introduzione a TeX/LaTeX}

Questo \`e un esempio di documento scritto con LaTeX.
Come si pu\`o vedere \`e gi\`a stato definito uno stile
generale del documento: article.

\subsection{Gli ambienti}

LaTeX utilizza gli ambienti per definire dei comportamenti
circoscritti a zone particolari del testo.
Per esempio, la centratura si ottiene utilizzando l'ambiente
center.

\begin{center}
Questo \`e un esempio di testo centrato.
\end{center}

% Fine del documento.
```

Markdown & sintassi wiki

```
An h1 header
============

Paragraphs are separated by a blank line.

2nd paragraph. *Italic*, **bold**, and `monospace`.
Itemized lists look like:

  * this one
  * that one
  * the other one
```

Formati testuali che usano trucchi testuali ad hoc per ottenere strutture e effetti tipografici precisi

Ottimi per testi lunghi, non permettono di catturare effetti tipografici specifici, forniscono supporto solo per le caratteristiche tipografiche più semplici.

JSON

JSON (*JavaScript Object Notation*) è un formato dati derivato dalla notazione usata da JS per gli oggetti.

```
{
  "nome":    ["Giuseppe", "Andrea", "Federico"],
  "cognome": "Rossi",
  "altezza": 180,
  "nascita": "1995-04-11T22:00:00.000Z",
  "indirizzo": {
     "via": {
        "strada": "Via Indipendenza",
        "numero": "15"
     },
     "citta": "Bologna",
     "nazione": "Italia"
  },
  "telefono": [
     { "tipo": "casa", "numero": "051 123456"},
     { "tipo": "cell", "numero": "335 987654"}
  ]
}
```

YAML

YAML Ain't a Markup Language è un'altra linearizzazione di strutture dati simile a JSON, ma con sintassi ispirata a Python:

- Superset di JSON (ogni file JSON è anche un file YAML)
- Indentazione come modello di annidamento
- Supporto di tipi scalari (stringhe, interi, float), liste (array) e hash (array associativi, mappe)
- supporto di stringhe multiriga con due delimitatori diversi
- Supporto di commenti

YAML: un esempio

```yaml
# con il carattere -
# si indicano delle liste
nome:
- Giuseppe
- Andrea
- Federico
cognome: Rossi

# oggetti annidati vogliono spazi
# in YAML non si usano tab
indirizzo:
  via:
    strada: "Via Indipendenza"
    numero: 15
  citta: "Bologna"
  nazione: "Italia"
telefono:
- tipo: "casa"
  numero: "051 123456"
- tipo: "cell"
  numero: "335 987654"
```

```yaml
# si supportano anche interi,
# float e date
altezza: 180
nascita: 1995-04-11T22:00:00.000Z

# con il carattere |
# i newline sono significativi
immagine: !!binary |
  R0lGODdhDQAIAIAAAAAAANn
  Z2SwAAAAADQAIAAACF4SDGQ
  ar3xxbJ9p0qa7R0YxwzaFME
  1IAADs=

# con il carattere <
# i newline sono irrilevanti
motto: <
  <blockquote>
    <p>Live and let live</p>
  </blockquote>
```

SGML

SGML (Standard Generalized Markup Language) è uno standard.

SGML è un meta-linguaggio non proprietario di markup descrittivo.

Facilita markup leggibili, generici, strutturali, gerarchici.

Linguaggio standard

SGML è uno standard ISO (International Standard Organization) n. 8879 del 1986. ISO è l'organizzazione mondiale degli standard, la più importante.

Essere uno standard per SGML significa che esso è il risultato di discussioni e compromessi di rappresentanti di tutte le comunità interessate, che è un investimento nel tempo di queste comunità, e che non dipende dai piani commerciali o dai capricci di una singola casa produttrice.

Meta-linguaggio di markup

Un meta-linguaggio è un linguaggio per definire linguaggi, una grammatica di costruzione di linguaggi.

SGML non è un linguaggio di markup, ma un linguaggio con cui definiamo linguaggi di markup.

SGML non dà dunque tutte le risposte di markup che possano sorgere a chi vuole arricchire testi per qualunque motivo, ma fornisce una sintassi per definire il linguaggio adatto.

SGML non sa cos'è un paragrafo, una lista, un titolo, ma fornisce una grammatica che ci permette di definirli.

Linguaggio non proprietario

Non proprietario significa che non esiste un'unica ditta o casa produttrice che ne detiene il controllo.

Viceversa RTF è © Microsoft, mentre PostScript e Acrobat sono © Adobe.

Essendo standard non proprietario, non dipende da un singolo programma, da una singola architettura. Gli stessi dati possono essere portate da un programma all'altro, da una piattaforma all'altra senza perdita di informazione.

Inoltre, anche l'evoluzione nel tempo è assicurata per lo stesso motivo.

Markup leggibile

In SGML il markup è posto in maniera leggibile a fianco degli elementi del testo a cui si riferiscono.

Essi possono sia essere usati da un programma, sia letti da un essere umano. Possono sia essere aggiunti da un programma (editor), sia da un essere umano con un programma non specifico.

Il markup in SGML è semplice testo facilmente interpretabile.

Markup descrittivo

Il markup in SGML non è pensato unicamente per la stampa su carta. E' possibile combinare markup utile per scopi o applicazioni diverse, ed in ogni contesto considerare o ignorare di volta in volta i markup non rilevanti.

Markup strutturato

SGML permette di definire delle strutture, suggerite o imposte, a cui i documenti si debbono adeguare.

Ad esempio, si può imporre che un testo sia diviso in capitoli, ognuno dei quali dotato di un titolo, una breve descrizione iniziale e almeno un paragrafo di contenuto.

È cioè possibile definire una serie di regole affinché il testo sia considerabile strutturalmente corretto.

Markup gerarchico

Le strutture imposte da SGML sono tipicamente a livelli di dettaglio successivi.

Gli elementi del testo possono comporsi gli uni con gli altri, permettendo di specificare la struttura in maniera gerarchica.

Ad esempio, si può imporre che il libro sia fatto di capitoli, e che questi a loro volta siano fatti di un una descrizione e molti paragrafi. Una descrizione sarà quindi fatta di elementi, ciascuno dei quali sarà una frase composta di testo; i paragrafi saranno testo eventualmente contenenti anche elementi in evidenza o figure.

I documenti SGML

Un documento in un linguaggio di markup definito sulla base di SGML è sempre composto delle seguenti tre parti:

- Dichiarazione SGML
- Dichiarazione di documento (DOCTYPE)
- Istanza del documento

Un esempio di SGML

SGML declaration

doctype

document instance

```
<!SGML "ISO 8879:1986"  ...>
<!DOCTYPE novel [
  <!ELEMENT novel      (front,content)>
  <!ELEMENT front      (title, subtitle?, author)>
  <!ELEMENT content    (chapter+)>
  <!ELEMENT chapter    (title, para+)>
  <!ATTLIST chapter    id CDATA #REQUIRED>
  <!ELEMENT title      #PCDATA>
  <!ELEMENT subtitle   #PCDATA>
  <!ELEMENT author     #PCDATA>
  <!ELEMENT para       #PCDATA>
]>
<novel>
  <front>
    <title>Three men in a boat</title>
    <subtitle>To say nothing of the dog!</subtitle>
    <author>Jerome K. Jerome</author>
  </front>
  <content>
    <chapter id="c1">
      <title>Chapter 1</title>
      <para>There were four of us ... </para>
      <para>We were all feeling ... </para>
    </chapter>
    <chapter id="c2">
      ...
    </chapter>
  </content>
</novel>
```

SGML Declaration

```
<!SGML "ISO 8879:1986" car. spec.>
```

La dichiarazione SGML contiene le istruzioni di partenza delle applicazioni SGML.

Essa permette di specificare valori fondamentali come la lunghezza dei nomi degli elementi, il set di caratteri usati, le specifiche caratteristiche di minimizzazione ammesse, ecc.)

Una dichiarazione SGML è lunga varie centinaia di righe.

Non è obbligatoria. Se è assente, viene usata una dichiarazione di default detta “Reference Concrete Syntax”.

La RCS definisce lunghezze e sintassi standard (come l'uso del carattere "<" per indicare l'inzio del tag).

Document Type Declaration

```
<!DOCTYPE nome [regole] >
```

La dichiarazione del tipo del documento serve a specificare le regole che permettono di verificare la correttezza strutturale di un documento.

Vengono cioè elencati [i file che contengono] gli elementi ammissibili, il contesto in cui possono apparire, ed altri eventuali vincoli strutturali.

Nella terminologia SGML, si parla di modellare una classe (cioè una collezione omogenea) di documenti attribuendogli un tipo.

La document instance

L’istanza del documento è quella parte del documento che contiene il testo vero e proprio, dotato del markup appropriato.

Esso contiene una collezione di elementi (tag), attributi, entità, PCDATA, commenti, ecc.

Le applicazioni SGML sono in grado di verificare se l’istanza del documento segue le regole specificate nel DTD, e di identificare le violazioni.

I componenti del markup

Un documento con markup di derivazione SGML (inclusi HTML, XML, ecc.) contiene una varietà dei seguenti componenti

- Elementi
- Attributi
- Entità
- Testo (detto anche `#PCDATA`)
- Commenti
- Processing Instructions

Elementi, Attributi, Entità, #PCDATA, Commenti, Processing Instructions

Gli elementi sono le parti di documento dotate di un senso proprio.

Il titolo, l'autore, i paragrafi del documento sono tutti elementi.

Un elemento è individuato da un tag iniziale, un contenuto ed un tag finale.

Non confondere i tag con gli elementi!

```
<titolo>Tre uomini in barca</titolo>
```

Elementi, Attributi, Entità, #PCDATA, Commenti, Processing Instructions

Gli attributi sono informazioni aggiuntive sull'elemento che non fanno effettivamente parte del contenuto (meta-informazioni).

Essi sono posti dentro al tag iniziale dell'elemento.

Tipicamente hanno la forma nome="valore"

```
<racconto tipo="romanzo">…</racconto>
<capitolo id="c1">Capitolo primo</capitolo>
```

Elementi, Attributi, Entità, #PCDATA, Commenti, Processing Instructions

Le entità sono frammenti di documento memorizzati separatamente e richiamabili all'interno del documento.

Esse permettono di riutilizzare lo stesso frammento in molte posizioni garantendo sempre l'esatta corrispondenza dei dati, e permettendo una loro modifica semplificata.

```
Oggi &egrave; una bella giornata.
Come dice &FV;: "divertitevi!"
Quanta felicit&#192;
```

Elementi, Attributi, Entità, #PCDATA, Commenti, Processing Instructions

Rappresenta il contenuto vero e proprio del documento.

Esso corrisponde alle parole, gli spazi e la punteggiatura che costituiscono il testo.

Viene anche detto #PCDATA (Parsed Character DATA) perché i linguaggi di markup definiscono character data (CDATA) il contenuto testuale vero e proprio, e quello degli elementi è soggetto ad azione di parsing (perlopiù per identificare e sostituire le entità).

Elementi, Attributi, Entità, #PCDATA, Commenti, Processing Instructions

I documenti di markup possono contenere commenti, ovvero note da un autore all'altro, da un editore all'altro, ecc.

Queste note non fanno parte del contenuto del documento, e le applicazioni di markup li ignorano.

Sono molto comodi per passare informazioni tra un autore e l'altro, o per trattenere informazioni per se stessi, nel caso le dimenticassimo.

```
<!-- Questo testo è ignorato dal parser -->
```

Elementi, Attributi, Entità, #PCDATA, Commenti, Processing Instructions

Le processing instructions (PI) sono elementi particolari (spesso di senso esplicitamente procedurale) posti dall'autore o dall'applicazione per dare ulteriori indicazioni su come gestire il documento XML nel caso specifico

Per esempio, in generale è l'applicazione a decidere quando cambiare pagina. Ma in alcuni casi può essere importante specificare un comando di cambio pagina (oppure tutti i cambi pagina di un documento già impaginato).

```
<?newpage?>
```

XML 1.0

Una raccomandazione W3C del 10 febbraio 1998.

È definita come un sottoinsieme di SGML

Molto più formalizzata della grammatica di SGML, usa una notazione formale, Extended Backus-Naur Form.

Documenti ben formati o validi

XML distingue due tipi di documenti rilevanti per le applicazioni XML: i documenti ***ben formati*** ed i documenti ***validi***.

In SGML, un DTD è necessario per la validazione del documento. Anche in XML, un documento è **valido** se presenta un DTD ed è possibile validarlo usando il DTD.

Tuttavia XML permette anche documenti **ben formati**, ovvero documenti che, pur essendo privi di DTD, presentano una struttura sufficientemente regolare e comprensibile da poter essere controllata.

Documenti XML ben formati

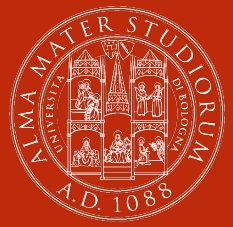
Un documento XML si dice ben formato se:

- Tutti i tag di apertura e chiusura corrispondono e sono ben annidati
- Esiste un elemento radice che contiene tutti gli altri
- Gli elementi vuoti (senza contenuto) utilizzano un simbolo speciale di fine tag: <vuoto/>
- Tutti gli attributi sono sempre racchiusi tra virgolette
- Tutte le entità sono definite.

Conclusioni

Qui abbiamo parlato di

- Importanza del markup nel testo
- Un po’ di storia del markup
  - Troff/Nroff
  - TeX e LaTeX
  - SGML – HTML
  - XML
  - JSON e YAML
  - Markdown e sintassi Wiki
- Cos'è SGML e di che cosa è fatto



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fabio Vitali

Corso di tecnologie web

Fabio.vitali@unibo.it