Il markup di documenti (II parte)

Fabio Vitali

Introduzione

Qui esaminiamo:

- Una brevissima storia del markup su computer
- Alcuni linguaggi di markup importanti
- SGML e gli elementi di markup tipici

La storia del markup (1)

Anni ‘60: primo WP in IBM

- Uomini e animali sono già nello spazio grazie ai computer, le aziende telefoniche ed elettriche già calcolano le bollette con i computer, eppure prima del 1964 nessuno aveva ancora trattato i testi con i computer.

GCA-GenCode e IBM-GML (1968-70)

- GenCode è il risultato della standardizzazione dei codici di tipografia (*Graphics Communications Association*)
- *Generalized Markup Language* di IBM è il linguaggio di markup per la documentazione interna e il prodotto BookMaster.

La storia del markup (2)

Anni ‘80: WP WYSIWYG e DTP: un passo indietro?

- Si diffondono i primi WP: WordStar, Word Perfect, MS Word
- Nasce il concetto di WYSIWYG (*What You See Is What You Get*): MacWrite e poi MS Word, ecc. L’enfasi è sulla verosimiglianza tra ciò che si vede sullo schermo e ciò che risulta sulla carta.
- Nascono i primi prodotti da editoria professionale (DTP: *Desk Top Publishing*): PageMaker, Ventura, Quark Xpress, ecc.

1986: SGML è standard ISO 8879

- Giudizi misti: “*Sounds Great, Maybe Later*”: manca il software, è considerato complicato e sofisticato.
- Nel 1988 il dip. della difesa americano (DoD) adotta SGML per l’iniziativa CALS (*Continuous Aided Logistic Support*)
- 1990: TEI (*Text Encoding Initiative*): un gruppo di umanisti generano linee guida per la strutturazione dei testi umanistici (prosa, teatro, poesia, ecc.).

La storia del markup (3)

1991: HTML

- Tim Berners Lee (CERN - Ginevra) inventa un sistema ipertestuale per la rete Internet chiamato “World Wide Web”. Il WWW è basato su tre protocolli, URI, HTTP e HTML
- HTML non nasce come un’applicazione SGML, ma solo come “ispirato” ad SGML. Solo in seguito verrà corretto per adeguarsi a SGML.

1997: la convergenza su XML

- Lo sviluppo e la correttezza degli standard connessi con il WWW sono gestiti dal W3C (World Wide Web Consortium).
- Nel 1995 la commissione sui linguaggi di markup decise di creare un nuovo linguaggio di markup con la completezza di SGML e la semplicità di HTML: *Extended Markup Language* (XML).
- Nel 1997 è uscito il primo standard per il linguaggio di markup (XML 1.0). In seguito i linguaggi connessi (XML-Namespaces, X-Pointer, X-Link, XSL, ecc.).

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 s
essere elaborato con Troff.
In questo caso, si presume che verr\
``\fBs\fP'' (con l'opzione \fB\-ms\fl

.ft B
1.1 Paragrafi

.ft P
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 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 L^AT_EX (1)

Realizzato agli inizi degli anni 80 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 arbitariamente complessi: matematica, chimica, musica, grafica vettoriale, grafica bitmap, ecc.

T_EX e L^AT_EX (2)

Il formatter TeX prende in input un documento TeX e genera un documento in formato DVI (DeVice Independent), che rappresenta una formulazione generica dell'aspetto grafico del documento.

Appositi programmi convertono poi il DVI in un formato utile per la stampante (ad esempio il PostScript).

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.

T_EX e L^AT_EX (3)

```
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.
```

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
- DTD
- Istanza del documento

Un esempio di SGML

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

```
<!DOCTYPE NOVEL [
  <!ELEMENT NOVEL    (FRONT,CONTENT) >
  <!ELEMENT FRONT (TITLE, SUBTITLE?, AUTHOR)>
  <!ELEMENT CONTENT (CHAPTER)+ >
  <!ELEMENT CHAPTER (TITLE, PARA+)>
  <!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>
      <TITLE>Chapter 1</TITLE>
      <PARA>There were four of us ... </PARA>
      <PARA>We were all feeling ... </PARA>
    </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 TIPO [markup] >
```

- 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"

```
<romanzo file="threemen.sgm">…</romanzo>
<capitolo N="1">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!"
```

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 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?>
```

Conclusioni

Qui abbiamo parlato di

- Importanza del markup nel testo
- Rilevanza del markup descrittivo per applicare scopi multipli allo stesso testo
- Qualche distinzione tra tipi di markup
- Un po’ di storia del markup
- Cos'è SGML e di che cosa è fatto

Riferimenti

- *J. H. Coombs, A. H. Renear, S. J. DeRose, Markup Systems and the future of Scholarly Text Processing, Communications of the ACM, 30(11), November 1987.*
- "I.2 A Gentle Introduction to SGML", in C.M. Sperberg-McQueen and L. Burnard (eds.), *Guidelines for Electronic Text Encoding and Interchange*, 1994, http://etext.virginia.edu/TEI.html
- E. Maler, J. El Andaloussi, *Developing SGML DTDs*, Prentice Hall, 1996