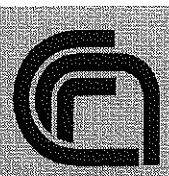


Consiglio Nazionale delle Ricerche

XML: un linguaggio estensibile per il World Wide Web

Massimo Martinelli

B4-04
feb-1999

Markup, Estensibilità

Markup è tutto ciò che ha un significato speciale, che deve essere ben caratterizzato: il testo scritto in grassetto, il testo sottolineato sono esempi di *markup*.
In XML tutto ciò che è compreso tra i caratteri "<" e ">" (*angled brackets*, parentesi angolari) è considerato *markup*, viene detto anche *tag* (etichetta), ad esempio:
<nome> è un *tag*.
XML è un metalinguaggio, contrariamente ad HTML che è un linguaggio predefinito, non ha *tag* predefiniti ma consente di definire nuovi metalinguaggi (esiste oggi la versione HTML in XML), è estensibile.
Anche HTML è un *markup language*, un linguaggio basato sui *markup*, ed è stato inizialmente definito in SGML. L'insieme delle regole di HTML sono contenute in un documento (separato dal file .html) chiamato DTD HTML (*Document Type Definition*).

Le componenti di XML

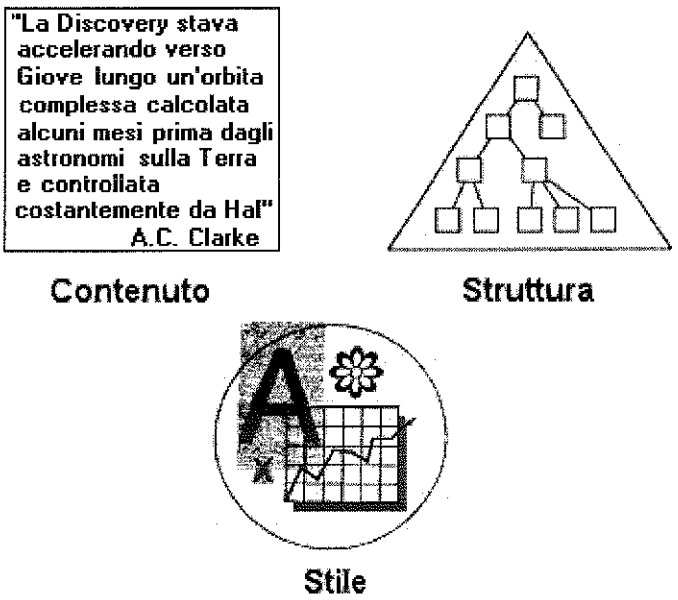
Uno dei problemi attuali più comuni è quello di scambiare documenti: ogni programma salva i propri dati in uno o più formati proprietari difficilmente scambiabili con altri programmi.
XML è stato studiato per consentire e facilitare scambi di dati anche tra applicazioni di tipo diverso, come ad esempio i database e i word processor (Oracle, Microsoft, Adobe prevedono di utilizzare il formato XML nelle prossime versioni dei loro programmi).
Per ottenere un documento facilmente interpretabile vi sono tre parti fondamentali che ogni documento dovrebbe tenere distinte:

- il contenuto;
- le specifiche relative agli elementi, la struttura (DTD);
- le specifiche relative alla visualizzazione, lo stile (*Stylesheet*).

Perché è importante tenere separate queste tre componenti: supponendo di dover cercare una informazione in un libro consultiamo l'indice, ove disponibile, altrimenti controlliamo i capitoli per vedere se l'argomento è inerente, quindi scorriamo i paragrafi, infine ci dedichiamo al contenuto.
La rappresentazione con un carattere diverso o un margine spostato più a destra ci aiutano a capire dove inizia un capitolo od un paragrafo e qui interviene lo stile che dovrebbe servire ad aiutare a leggere meglio, a capire la relazione della struttura con il contenuto.

E' ovvio che se modifichiamo il linguaggio e intendiamo farlo conoscere ad altri o lo vogliamo utilizzare come formato di scambio abbiamo la necessità di utilizzare una soluzione aperta non proprietaria, in sostanza dobbiamo dichiarare il significato delle estensioni che abbiamo effettuato, rendendo pubblico il DTD.

Figura 1 - Le tre componenti di un documento



Come si presenta un documento XML

Uno degli obiettivi di progettazione di XML è che sia in un formato leggibile dall'uomo, per intendersi non può essere in formato binario, e dovrebbe essere ragionevolmente chiaro. Nell'esempio di figura 2, biblioteca.xml, vediamo come può essere scritto un documento XML:

Figura 2 - biblioteca.xml

```
<?xml version="1.0"?>
<biblioteca>
    <libro codice="R414">
        <titolo>2001: Odissea nello spazio</titolo>
        <autore>
            <cognome>Clarke</cognome>
            <nome>Arthur Charles</nome>
        </autore>
        <editore>Rizzoli</editore>
        <parola_chiave>romanzo</parola_chiave>
        <parola_chiave>fantascienza </parola_chiave>
    </libro>
</biblioteca>
```

Ogni documento XML inizia con un prologo che contiene una dichiarazione di versione

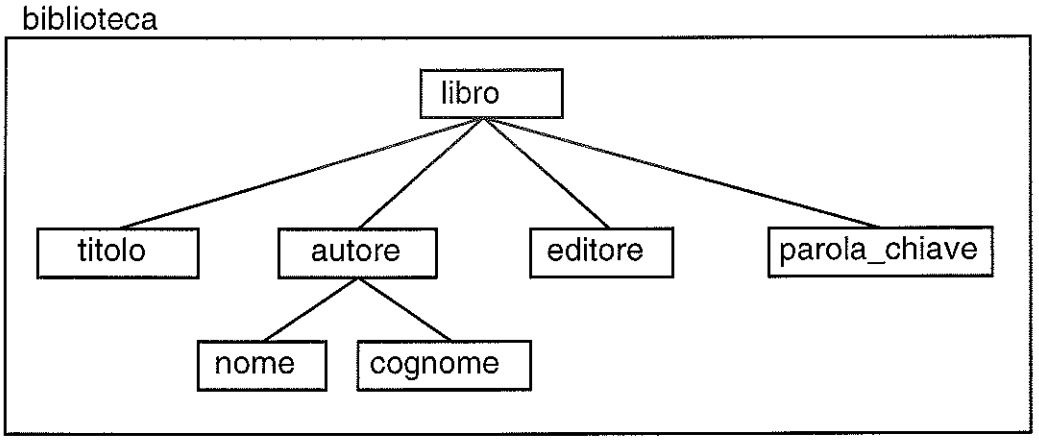
```
<?xml version="1.0"?>
```

Ogni *tag* di apertura deve avere un corrispondente *tag* di chiusura; se l'elemento non ha contenuto, come nel caso di BR in HTML, invece di
</BR> è consentito l'uso della forma più concisa
.
Le maiuscole e le minuscole sono interpretate differentemente, pertanto il *tag* <nome> è diverso da <Nome> e da <NOME>; per convenzione i *tag* HTML si scrivono in maiuscolo, quelli XML in minuscolo.
Alcuni caratteri e sequenze di caratteri sono riservati, pertanto non si possono utilizzare nei nomi di *tag* (%, xml, ...).

DTD - *Document Type Definition*

Il DTD contiene le regole di definizione dei *tag*, indica gli elementi e il loro ordine all'interno del documento XML, contrariamente a SGML il suo uso non è obbligatorio, poiché XML ha regole più vincolanti, ne è comunque consigliato l'utilizzo per verificare la validità, la congruità del documento. Il DTD può essere interno o esterno al documento XML, il suo nome per convenzione corrisponde a quello dell'elemento radice (nel nostro esempio "biblioteca").
Supponiamo di avere la seguente struttura gerarchica che rappresenti la nostra ipotetica biblioteca:

Figura 3 - Struttura gerarchica



Nella prossima figura (figura 4) è mostrato un DTD XML, biblioteca.dtd, che rappresenta la struttura definita precedentemente:

Figura 4 - biblioteca.dtd

```
<!DOCTYPE biblioteca [
<!ELEMENT biblioteca (libro+)>
<!ELEMENT libro (titolo, autore+, editore, parola_chiave+)>
<!ATTLIST libro
codice ID #REQUIRED>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT autore (cognome, nome)>
<!ELEMENT editore (#PCDATA)>
<!ELEMENT parola_chiave (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT nome (#PCDATA)>
]>
```

Il DTD, se è interno al documento XML, inizia con una dichiarazione DOCTYPE:
`<!DOCTYPE elemento_radice [elemento, attributo, entità, notazione]>`
Nel caso il DTD sia esterno la dichiarazione DOCTYPE va fatta all'interno del documento XML, il file biblioteca.xml sarebbe cosi scritto:

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

Successivamente alla dichiarazione DOCTYPE si dichiarano gli elementi:
`<!ELEMENT nome_elemento (nomi_degli_elementi_permessi)>`
Nel nostro esempio, osservando la dichiarazione dell'elemento "libro" si può notare che è composto dagli elementi "titolo", "autore", "editore" e "parola_chiave": questi elementi sono separati da una virgola pertanto devono essere presenti nel file biblioteca.xml in questo stesso ordine.
Prendiamo in esame le seguenti dichiarazioni:

```
<!ELEMENT persona (nome|cognome)>
<!ELEMENT persona (nome|cognome|email)*>
```

Nel primo caso il carattere "|" sta a significare che l'elemento "persona" sarà costituito dall'elemento "nome" oppure dall'elemento "cognome"; nel secondo caso, in cui è presente un carattere "*", da 0, 1 o più elementi "nome", "cognome", "email" in qualsiasi ordine.
Si può notare che l'elemento autore è seguito dal carattere "+", questo significa che di elementi autore ce ne sarà almeno uno ma che ce ne potrebbero essere più di uno.

Se un elemento, o un gruppo di elementi elencati tra parentesi, è seguito dal carattere "?" potrà essere presente zero o una volta.
<!ELEMENT titolo (#PCDATA)> significa che l'elemento titolo potrà essere composto da qualsiasi testo o altro carattere che non sia un markup o ", & oppure]] (PCDATA = *Parsed Character Data*).

Mediante l'uso di attributi è possibile specificare gli elementi: si dichiarano nella forma:

```
<!ATTRIBUTE nome_elemento
nome_attributo tipo_di_attributo valore_di_default>
```

Nel nostro esempio ogni elemento libro avrà un codice univoco (ID).
I tipi di attributo possono essere CDATA (qualsiasi testo), ID o una lista di valori (tipo enumerato) e si dichiarano in questo modo:

```
<!ATTLIST nome_elemento
nome_attributo tipo_attributo valore_di_default>
```

Il valore di un attributo è obbligatorio quando è specificata l'opzione #REQUIRED (nel nostro esempio il codice dell'elemento libro è obbligatorio), può non avere un codice specificato se l'opzione è #IMPLIED, oppure il solo valore specificato è valido con l'opzione #FIXED.

Oltre agli elementi e agli attributi il DTD può contenere entità e notazioni.
Si possono definire entità nel caso ci siano elementi ripetuti spesso: ad esempio

```
<!ENTITY XML "eXtensible Markup Language">
```

ogni occorrenza di &XML sarà sostituita in fase di visualizzazione dalla stringa " eXtensible Markup Language ".

Le entità si possono utilizzare per rappresentare caratteri riservati come ad esempio "<" o ">" (&lt, &gt), o possono essere usate per far riferimento a documenti esterni nel caso il documento XML non sia composto da un unico file:

```
<!ENTITY introduzione SYSTEM "introduzione.txt">
```

Possono esserci anche entità parametriche

```
<!ENTITY % [nome] "[nomi]">
```

ad esempio:

```
<!ENTITY % headings "H1 | H2 | H3 | H4">
<!ENTITY BODY (%headings | P | DIV | BR)*>
```

Le dichiarazioni di notazione servono per identificare specifici tipi di dati binari esterni, come ad esempio i file in formato "gif"

```
<!NOTATION GIF87A SYSTEM "GIF">
```

Perché un documento si possa dire "ben formato" deve:

- contenere almeno un elemento;
- esistere un *tag* unico di apertura e di chiusura che contenga l'intero documento; questo *tag* è chiamato elemento radice (*root*);
- tutti i tag devono essere nidificati (<B><I>testo</B></I> contrariamente a HTML non è consentita);
- tutte le entità devono essere dichiarate.

Un documento si dice "valido" quando contiene un DTD e rispetta le regole definite nel DTD.

Stylesheet

Con lo stile separato dal contenuto e dalla struttura non sarà più necessario riscrivere tutto il documento quando si cambia la presentazione, basterà riscrivere solo gli elementi necessari per la formattazione che, anziché preparati per una specifica visualizzazione su web, saranno modificati per inviare l'output ad altri dispositivi, ad esempio ad un sintetizzatore vocale, su un foglio di carta A3, o semplicemente per presentare su web in modo differente, o elementi diversi.
Le specifiche di XML non fanno riferimento ad alcun metodo di stile da applicare, pertanto per effettuare la visualizzazione su Web bisogna utilizzare CSS (*Cascade Style Sheet*), trasformare l'output del documento in HTML oppure usare formati proprietari che ovviamente sono applicabili solo su piattaforme e da programmi specifici.
I browser HTML interpretano a modo loro i *tag* per la visualizzazione o addirittura utilizzano *tag* proprietari, questo comporta un output differente.
Anche se la versione 2 di CSS (le specifiche sono disponibili all'indirizzo http://www.w3.org/TR/REC-CSS2) ha aggiunto nuove possibilità, l'attuale *Stylesheet* risulta limitato in quanto non consente modifiche al documento.
Per superare questi limiti sono allo studio nuovi stili per XML, in particolare XSL (*eXtensible Stylesheet Language*) è un linguaggio di stile basato sul DSSL (*Document Style Semantics and Specification Language*), quest'ultimo è utilizzato in particolar modo con i documenti SGML.
Le potenzialità di XSL, nettamente superiori a CSS, unitamente alla semplicità, fanno ritenere che questo possa essere lo stile di riferimento per XML.

Una versione *draft* di XSL è disponibile all'indirizzo:
http://www.w3.org/TR/1998/WD-xsl-19980818

Nelle figure 5 e 6 un foglio di stile CSS per biblioteca.xml, biblioteca.css, e la visualizzazione del documento biblioteca.xml con Internet Explorer 5 beta 2:

Figura 5 - biblioteca.css

```
titolo { display: block;
         text-align: center;
         background: blue;
         color: white;
         font-family: Arial;
         font-size: 20pt
       }
autore { display: block;
         margin-left: 10%;
         text-align: left;
         color: red;
         font-family: Arial;
         font-style: italic;
         font-size: 14pt
       }
cognome, nome { display: inline;
              }
editore { display: block;
          margin-left: 15%;
          color: green;
          font-family: Arial;
          font-size: 14pt
        }
parola_chiave { display: block;
                margin-left: 5%;
                color: black;
                font-family: "Times New Roman";
                text-align: justify
                font-size: 14pt
              }
```

Figura 6 - Visualizzazione di biblioteca.xml con lo stile CSS

XLL - eXtensible Linking Language

Una delle caratteristiche fondamentali che hanno permesso lo sviluppo di Internet ed in particolar modo degli ipertesti è il *link*.
Attualmente HTML implementa solo link di tipo unidirezionale, SGML non supporta link di *markup*.
E' in fase di sviluppo un linguaggio, XLL (*eXtensible Linking Language*), che fornisce ad XML collegamenti che non sono solo del tipo unidirezionale.
XLL si sviluppa in due direzioni: Xlink e Xpointer.

Xlink (*XML Link Language*) consente collegamenti tra più di due risorse, tramite questi *link* è possibile indicare se la risorsa collegata deve essere inserita in un punto preciso del documento corrente (in HTML questo si poteva fare solo con i *frame*), se deve prendere il posto del documento corrente o se deve essere aperta una nuova finestra. E' possibile utilizzare link bidirezionali mediante i quali si attivano flussi di informazioni tra due risorse, oppure con un singolo link si può scegliere più di una destinazione (link multiplo); inoltre è possibile che il *link* sia attivato automaticamente o manualmente (usualmente quando l'utente preme il bottone del mouse sul testo o sull'immagine associata al *link*).
La bozza del documento del W3C su XLink è consultabile all'indirizzo
http://www.w3.org/TR/1998/WD-xlink-19980303

XPointer (*XML Pointer Language*) permette di collegarsi ad un punto di un documento anche se non è stato referenziato. In HTML questo è possibile solo se l'autore del documento ha inserito un ancora (*anchor*), con XPointer è sempre possibile poiché è semplice localizzare un qualsiasi punto di un documento mediante la struttura dello stesso o riferirsi ad esso a seconda del contesto ad esempio sarà possibile riferirsi al quarto elemento del primo figlio "nome" del documento mediante la seguente notazione: `child(1,nome).child(4,#element)`
La versione *draft* del documento del W3C su XPointer è consultabile all'indirizzo
http://www.w3.org/TR/1998/WD-xptr-19980303

DOM - Document Object Model

Lo scopo del DOM (*Document Object Model*) è quello di definire una interfaccia, indipendente dal linguaggio, che permetta ai programmi di accedere dinamicamente e di modificare il contenuto, la struttura e lo stile di un documento; è possibile applicarlo sia ai documenti XML sia a quelli HTML.
Attraverso il DOM gli elementi del documento XML sono visti come nodi di un albero (i nodi non rappresentano una struttura, bensì degli oggetti) o più precisamente di una foresta che può contenere più di un albero, sono quindi offerti i metodi necessari per manipolarli in questa forma.
Il livello 1 del DOM è suddiviso in due parti: Core e HTML.
La prima parte definisce un insieme di interfacce di basso livello per rappresentare documenti strutturati, il nucleo, ma anche di livello più alto per rappresentare documenti XML. La seconda parte definisce un insieme di interfacce, che usano quelle definite nel Core, per avere un accesso semplice ai documenti HTML.
Il DOM offre strumenti di interrogazione sugli elementi e sugli attributi e un modello per la gestione degli eventi che possono essere generati da qualsiasi elemento.
Sarà previsto un meccanismo per la gestione degli errori, oltre a meccanismi di sicurezza che evolveranno con i vari livelli del DOM.
Le specifiche di livello 1 sono consultabili al seguente indirizzo:
http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001

Alcune applicazioni

I nuovi linguaggi definiti con XML sono detti applicazioni: nonostante le specifiche del linguaggio esistano da pochi mesi ci sono già numerose applicazioni.
Alcuni esempi:
Mathematical Markup Language (MathML) definisce un linguaggio per la matematica, *Chemical Markup Language* (CML) definisce un linguaggio per la chimica, *Channel Definition Format* (CDF) utilizzato come formato aperto per scambiare informazioni sui canali, *Resource Description Framework* (RDF) strumento per descrivere i metadati, *Open Software Description* (OSD) utilizzato per descrivere il software, *Sincronized Multimedia Integration Language* (SMIL) utilizzato per descrivere elementi multimediali.
Le applicazioni appena elencate sono solo alcuni DTD pubblici, standardizzate ma estensibili, universalmente riconosciute nei relativi settori di utilizzo, per la maggior parte sono derivate da DTD usati in SGML, nuove applicazioni saranno presto disponibili per molte discipline.

Il software

Un programma per la elaborazione di documenti XML è costituito da un "*parser*" (analizzatore) e da un *browser*. Il *parser* controlla se il documento è conforme al DTD (se il documento è "ben formato") e, se validante, la sua validità (documento "valido"); il *browser* si occupa di visualizzare il documento sulla base dello *stylesheet.*
Nella prossima tabella sono riportati alcuni programmi divisi per tipologia:

Parser validanti (verificano la validità del codice),	MSXML (Microsoft), XML for Java (IBM), DXP (DataChannel)
Parser non validanti (verificano solo se il documento è ben formato)	Lark (Tim Bray), XP (James Clark)
Interfacce ai parser:	Sax, DOM scritte in Java, Perl, Python, Frontier
Editor:	Xmetal(annunciato), Framemaker (Adobe), Adept (ArborText), Balise (Chrystal), Word (annunciato, ma proprietario),
Database:	Oracle (annunciato), Access (annunciato)
Browser:	IE5 beta 2, Netscape 5 beta, Jumbo (usato per CML)

Dove reperire altre informazioni

Poiché XML è un linguaggio nuovo ed in evoluzione i libri in commercio rischiano di essere obsoleti già al momento della pubblicazione o comunque possono trattare informazioni non più valide, pertanto la documentazione aggiornata è da ricercare direttamente su Web: il successivo elenco di indirizzi può costituire un buon punto di partenza:

La home page di XML sul sito del W3C: http://www.w3.org/XML
Le faq gestite da alcuni membri dell'XML Working Group : http://www.ucc.ie/xml
La pagina dell'SGML/XML group: http://www.oasis-open.org/cover/sgml-xml.html
Il sito di Microsoft su XML: http://www.microsoft.com/xml
Il sito di Netscape: http://www.mozilla.org
La pagina della GCA (Graphics Communication Association) su XML:
http://www.gca.org/conf/xml/xml_what.htm
Il sito della ArborText: http://www.arbortext.com/xml.html
L'università di XML: http://www.xmlu.com

Il sito XML-IT, contenente anche una mail-list sull'argomento: http://www.xml.it

Conclusioni: perché utilizzare XML

Queste, in sintesi, le motivazioni per scegliere XML:

- consente di utilizzare documenti strutturati;
- è estensibile, permette di aggiungere sempre nuovi marcatori;
- offre un ottimo formato di scambio di dati, inoltre è un formato che probabilmente durerà a lungo poiché strutturato, estensibile, non ambiguo e completamente leggibile (non binario) e sarà comunque riutilizzabile, considerando anche che i programmi ad ogni nuova versione cambiano formato;
- la strutturazione e l'utilizzo di un linguaggio estensibile basato su *tag* consente una più semplice interazione con altri programmi, compresi i data base, e quindi un trattamento dei dati più semplice ed efficace;
- i *link* offrono nuove possibilità;
- portabilità (indipendente dalla piattaforma e dal processore;
- permette un semplice utilizzo di metadati, come Dublin Core, Warwick Framework, RDF;
- ricerche più semplici e più efficaci, prendiamo ad esempio una interrogazione effettuata tramite un motore di ricerca: attraverso il controllo sui *tag* sarà più inerente a ciò che realmente stiamo cercando;
- offre un buon meccanismo di rappresentazione, una ottima capacità di rappresentare dati complessi (notazioni matematiche, interfacce grafiche);
- offre possibilità di presentazioni superiori a quelle di HTML, per ottenere risultati simili con HTML è necessario utilizzare Javascript, Java o altri linguaggi;
- è semplice ma potente.

Bibliografia

1) "Extensible Markup Language (XML)" http://www.w3.org/XML

2) XML faq: http://www.ucc.ie/xml/faq.txt

3) "A Technical Introdution to XML" - Norman Walsh. http://www.xml.com

4) "XML Tutorial" - Frank Boumphrey.
http://www.hypermedic.com/style/xml/index.html

5) "XML Tutorials for Programmers"
http://www.software.ibm.com/xml/education/tutorial-prog/abstract.html

6) "Introducing the Extensible Markup Language (XML)" - Robin Cover
http://www.oasis-open.org/cover/xmlIntro.html

7) "XML: Structuring Data for the Web" - Ken Sall
http://www.wdvl.com/Authoring/Languages/XML/Intro/index.html

8) "XML for Dummies" - Ed Tittel, Norbert Mikula, Ramesh Chandak. Apogeo.
IDG Books Woldwide

9) "La seconda rivoluzione" Internet News Luglio Agosto 1998