XML \ SGML

Fabio Vitali

3 marzo 2000

Introduzione

Qui esaminiamo in breve tutti gli aspetti di XML che non sono derivati da SGML:

- Differenze sintattiche
- Differenze architettoniche
- Usi innovativi e visioni del futuro di XML

XML

XML (Extensible Markup Language [sic!]) è un meta-linguaggio di markup, progettato per lo scambio e la interusabilità di documenti strutturati su Internet.

XML prevede una sintassi semplificata rispetto a SGML, e definisce contemporaneamente una serie piuttosto lunga di linguaggi associati: uno per i link, uno per i nomi di tag, uno per i fogli di stile, uno per la descrizione di meta-informazioni, ecc.

XML si propone di integrare, arricchire e, nel lungo periodo, sostituire HTML come linguaggio di markup standard per il World Wide Web.

Perché XML?

HTML nacque come un DTD di SGML (non proprio!!!), che permetteva di mettere in rete documenti di un tipo molto specifico, semplici documenti di testo con qualche immagine e dei link ipertestuali.

Con il successo del WWW, HTML venne iniziato ad usare per molti scopi, molti più di quelli per cui era stato progettato.

Si iniziò ad abusare dei tag di HTML per gli effetti grafici che forniva, più che per gli aspetti strutturali o semantici.

Si iniziarono a desiderare elaborazioni sofisticate sui dati HTML, elaborazioni che non era possibile fornire.

Si iniziò a trovare limitata la capacità grafica di HTML, anche abusando dei tag.

Perché non SGML?

SGML ha molti pregi, ma ha dalla sua una complessità d'uso e di comprensione notevole. Inoltre, a SGML mancano caratteristiche di notevole importanza per l'uso pratico, come link ipertestuali e specifiche grafiche.

L'avvento di HTML ha fatto capire come i linguaggi di markup siano ormai maturi per essere compresi dal largo pubblico, ma che la semplicità d'uso di HTML doveva costituire un elemento di partenza.

XML contiene tutte le caratteristiche di SGML che servono per creare applicazioni generali senza scendere nel livello di dettaglio e pedanteria richiesti da SGML.

I vantaggi di XML (1)

Documenti auto-descrittivi

- La scelta dei nomi degli elementi può essere fatta per facilitare la comprensione del ruolo strutturale dell'elemento.
- Inoltre, l'uso di un DTD può esplicitare le regole di composizione ed i rapporti possibili tra le varie parti dei documenti.

Struttura navigabile dei documenti

- La rigida struttura ad albero e l'assenza di regole di minimizzazione rendono semplice la visualizzazione e l'analisi della struttura del documento, e la possibilità di visualizzare il documento è indipendente dal foglio di stile che vi si applica.

I vantaggi di XML (2)

Platform-independence

- XML è uno standard aperto, e chiunque può realizzare strumenti che lo usino come formato di dati.

Facile convertibilità a formati Web

- La totale interdipendenza tra XML, SGML, HTML etc. fa sì che la conversione tra formati interni e formati per il Web sia facile.

Cosa si fa con XML? (*Bradley*)

Data Interchange

- Ogni volta che più programmi si debbono scambiare dati, ci sono problemi di compatibilità. Ogni programma ha le proprie assunzioni in termini di caratteri, separatori, ripetibilità di elementi, differenza tra elementi vuoti e assenti, ecc.
- XML si propone come la sintassi intermedia più semplice per esprimere dati anche complessi in forma indipendente dall'applicazione che li ha creati.

Document publishing

- XML è ideale come linguaggio per esprimere documenti strutturati o semi strutturati, e per esprimerli in maniera indipendente dalla loro destinazione finale.
- Lo stesso documento XML può essere preso e trasformato per la stampa, il Web, il telefonino, l'autoradio.

Cosa si fa con XML? Bosak (1)

- Applicazioni che richiedono che il client Web si ponga a mediare tra due o più database eterogenei
- Applicazioni che distribuiscono una parte significativa del carico computazionale dal server al client
- Applicazioni che richiedono che il client Web presenti view diverse degli stessi dati agli utenti
- Applicazioni in cui agenti Web intelligenti adattano la scoperta di informazioni alle esigenze degli specifici utenti.

Da J. Bosak, ***XML, Java, and the future of the Web***,
`http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm`

Cosa si fa con XML? *Bosak* (2)

Accesso a database eterogenei

- Ogni volta che è necessario trasferire dei dati da un database all'altro, la soluzione più economica a tutt'oggi è stampare i dati dal primo DB su carta e ribatterli a mano sul secondo.
- Idealmente io vorrei accedere via Web ai dati del primo DB, selezionare quelli che voglio in una cartella, e sbattere la cartella sul secondo DB, che si preoccupa di adattarli alle sue esigenze.
- Il secondo DB, dunque, deve essere in grado di comprendere la sintassi dei dati, di interpretare la struttura (eventualmente, in parte, aiutato da un essere umano) e di isolare le informazioni di suo interesse.
- Per questo potrebbe essere aiutato da un formato di interscambio tipo XML, che permetterebbe di etichettare i dati esplicitamente ed in maniera generale e comprensibile agli esseri umani.

Cosa si fa con XML? *Bosak* (3)

Computazioni client-side

- Esistono molte esigenze di testing e computazione su oggetti descrivibili parametricamente:
  - Caratteristiche e funzionalità di chip, semilavorati, e prodotti industriali
  - Scheduling in aerei, treni, ecc.
  - Shopping on-demand, e user-tailoring
  - Applicazioni per il customer suppot
- In tutti questi casi, attualmente si creano applicazioni server-side che interrogano i database per i parametri e usano cicli del server per le computazioni, mentre i client sono in attesa.
- Poter esprimere in Java o altri linguaggi client-side la logica della computazione, che scarica i parametri dal sito giusto ed esegue le computazioni indipendentemente, sarebbe molto comodo, e permetterebbe confronti incrociati e ogni altro tipo di valutazione ottimale per le esigenze di chi compra.

Cosa si fa con XML? *Bosak* (4)

Viste selettive

- L'esempio tipico è l'indice sommario dinamico di un documento: interrogo una base documentaria e ottengo il primo livello di indice di un documento. Seleziono una voce e ri-interrogo la base dati per avere il secondo livello dell'indice.
- Ogni espansione richiede un passaggio al server, con ovvi problemi di latenza. Sarebbe possibile fare tutto client-side con Javascript, ma o si fa l'indice a mano del documento HTML, oppure bisogna ricorrere a documenti ben strutturati, come XML.
- Altri esempi:
  - Un grafico che si trasforma in una tabella
  - Un documento annotato in cui vedo il contenuto, o le annotazioni, o tutti e due
  - Un manuale di due versioni dello stesso sistema, con testi e immagini che cambiano a seconda di quale specifica versione si sta esaminando.

Cosa si fa con XML? *Bosak* (5)

Agenti Web

- Mattew Fuchs (Disney Imagineering): “Data needs to know about itself, and data needs to know about me”
- Agenti di filtro, selezione, rilevamento hanno bisogno di sapere le caratteristiche dei dati che stanno filtrando in maniera vendor-independent, ben strutturata e flessibile (nuove esigenze, categorie, comunità virtuali, sub-società si formano continuamente)
- Ad esempio, bot personalizzati, la guida dei canali TV, i sistemi di classificazione del contenuto delle pagine Web, ecc.
- Su questo specifico tema esistono argomenti di tesi di laurea.

Un esempio: XMLNews (1)

XMLNews definisce il contenuto testuale e le meta-informazioni di notizie da agenzia stampa. E' una parte dello standard denominato ***News Industry Text Format*** (NITF), sviluppato dal ***International Press Telecommunications Council*** e dalla ***Newspaper Association of America***.

XMLNews è composto di due parti:

- XMLNews-Story è un DTD XML per descrivere in maniera variamente arricchita il testo delle notizie
- XMLNews-Meta definisce il formato delle meta-informazioni per notizie d'agenzia. E' conforme al Resource Description Framework (RDF), e on si riferisce solo alle notizie testuali, ma anche a immagini, video-clip, ecc.

Un esempio: XMLNews (2)

XMLNews-Story: il testo di una notizia di agenzia è diviso in tre parti: l'head contiene informazioni di organizzazione, mentre il body è a sua volta diviso in intestazione e contenuto.

```
<?xml version="1.0"?>
<nitf>
  <head> <title>Colombia Earthquake</title> </head>
  <body>
    <body.head>
      <hedline><hl1>143 Dead in Earthquake</hl1></hedline>
      <byline><bytag>By Jared Kotler, AP </bytag></byline>
      <dateline>
        <location>Bogota, Colombia</location>
        <story.date>January 25 1999 7:28 ET</story.date>
      </dateline>
    </body.head>
    <body.content> … </body.content>
  </body>
</nitf>
```


Un esempio: XMLNews (3)

XMLNews-Story: Il body ha un markup minimale di struttura del testo:

```
<?xml version="1.0"?>
<nitf> <head> … </head> <body> <body.head> … </body.head>
    <body.content>
      <p>An earthquake struck western Colombia on Monday,
         killing at least 143 people and injuring more than
         900 as it toppled buildings across the country's
         coffee-growing heartland, civil defense officials
         said.
      </p>
      <p>The early afternoon quake had a preliminary
         magnitude of 6, according to the U.S. Geological
         Survey in Golden, Colo. Its epicenter was located
         in western Valle del Cauca state, 140 miles west
         of the capital, Bogota.
      </p>
    </body.content> </body>
</nitf>
```


Un esempio: XMLNews (4)

XMLNews-Story: Però è possibile in qualunque momento aggiungere informazioni inline:

```
<p>An <event>earthquake</event> struck <location>western
<country>Colombia</country></location> on <chron
norm="19990125">Monday</chron>, killing at least 143
people and injuring more than 900 as it toppled buildings
across the country's coffee-growing heartland,
<function>civil defense officials</function> said.</p>
```

Questo permette di arricchire la storia con molte informazioni e in maniera semi-automatica:

- **Nella ricerca**: è possibile cercare tutto quello che è successo in Colombia, o cosa è successo in una certa data.
- **Nella presentazione**: un provider potrebbe fornire semi-automaticamente dei link o delle cartine della Colombia.
- **Nell'organizzazione delle news**: è possibile cercare tutti i terremoti effettivi, e non le notizie che ne usano la parola, magari figurativamente.

Un esempio: XMLNews (5)

XMLNews-Meta: Assieme ad ogni notizia, vengono scritte delle informazioni sulla notizia, che possono avere una distribuzione separata.

XMLNews-Meta permette di gestire insieme informazioni come:

- Informazioni sul contenuto della notizia (titolo, lingua, formato, ecc.)
- Informazioni sulle date della notizia: creazione, pubblicazione, scadenza, ecc.
- Informazioni sulla provenienza ed attendibilità della notizia
- Informazioni sui possessori dei diritti di distribuzione e copyright
- Informazioni di classificazione ed organizzazione
- Link a documenti connessi: versioni precedenti, seguenti, ed altre notizie connesse.

Cosa c’è con XML?

XML è in realtà una famiglia di linguaggi, alcuni già definiti, altri in corso di completamento. Alcuni hanno l’ambizione di standard, altri sono solo proposte di privati o industrie interessate. Alcuni hanno scopi generali, altri sono applicazioni specifiche per ambiti più ristretti.

Noi di occupiamo, tra gli altri, di:

- XML 1.0: un meta-linguaggio di markup, sottoinsieme di SGML
- XML-Namespace: un meccanismo per la convivenza di nomi di tag appartenenti a DTD diversi
- XPath, XPointer e XLink: tre linguaggi per la creazione di link ipertestuali
- XSL e XSLT: due linguaggi di stylesheet per XML
- XML schema: un linguaggio per la specifica di criteri di validazione di documenti XML
- RDF: un linguaggio per l’espressione di metainformazioni su documenti XML.

XML 1.0

- Una raccomandazione W3C del 10 febbraio 1998.
- È definita come un sottoinsieme di SGML
- URL ufficiale: http://www.w3.org/TR/REC-xml
- Traduzione ufficiale in italiano:
  `http://www.iat.cnr.it/xml/REC-xml-19980210-it.html`
- Molto più formalizzata della grammatica di SGML, usa una notazione formale, ***Extended Backus-Naur Form***.

Criteri di progettazione di XML (1)

Nel documento ufficiale di XML si elencano i seguenti obiettivi progettuali di XML:

1. XML deve essere utilizzabile in modo diretto su Internet.
   - Non significa che deve essere possibile usarlo sul browser del giorno.
   - Significa che si dovevano tenere in conto le esigenze di applicazioni distribuite su reti a larga scala.
2. XML deve supportare un gran numero di applicazioni.
   - Cioè XML non si limita al supporto di documenti in rete, ma a una larga classe di applicazioni che non c'entrano con la rete. Specificamente: deve essere possibile creare applicazioni come tool di authoring, filtri, formattatori, e traduttori.

Criteri di progettazione di XML (2)

3. XML deve essere compatibile con SGML
   - Tool SGML esistente debbono essere in grado di leggere e scrivere documenti XML
   - Istanze XML debbono essere istanze SGML così come sono, senza traduzioni, per quanto semplici.
   - Dato un documento XML, deve essere possibile generare un DTD SGML tale per cui un tool SGML esegue lo stesso parsing di un tool XML.
   - XML deve avere essenzialmente lo stesso potere espressivo di SGML.

Questi goal sono stati sostanzialmente raggiunti.

Criteri di progettazione di XML (3)

4. Deve essere facile lo sviluppo di programmi che elaborino documenti XML
   - Deve essere possibile creare applicazioni XML utili che non dipendano dal leggere ed interpretare il DTD
   - Obiettivo dichiarato: un diplomato in informatica deve essere in grado di scrivere un processore minimale XML in meno di una settimana.
5. Il numero di caratteristiche opzionali deve essere mantenuto al minimo possibile, idealmente a zero.
   - SGML, per generalità, aveva adottato un numero molto alto di caratteristiche opzionali, di dubbia utilità, o molto specifiche
   - Risultato: ogni processore SGML implementava solo una parte delle caratteristiche opzionali, e quindi documenti SGML conformi che potevano essere letti da un processore SGML non venivano letti da un altro, e viceversa.

Criteri di progettazione di XML (4)

6. I documenti XML dovrebbero essere leggibili da umani e ragionevolmente chiari.
   - Formati testuali sono più aperti, più utili, più gradevoli da lavorarci che formati binari.
   - Inoltre, per quanti capricci possa fare il tuo editor specializzato XML, puoi sempre aprire il documento con un editor di testi e rimettere a posto le cose.
7. La specifica del linguaggio XML deve avvenire rapidamente.
   - La paura era che le esigenze di estensibilità del Web potessero essere soddisfatte da una qualche combinazione di complicati formati binari e di accrocchi proprietari.
     Es: DHTML!

Criteri di progettazione di XML (5)

8. La progettazione XML deve essere formale e concisa.
   - La specifica di SGML è composta di un documento di oltre 300 pagine in testo, ottuso e burocratico. Il manuale SGML ne richiede più di 600, e comunque non è leggibile.
   - Inoltre non è neanche immediatamente utilizzabile da un programmatore per realizzare tool.
   - La scelta di formalismi nitidi e pochi commenti ha permesso la creazione di una specifica notevolmente più corta e immediatamente utilizzabile dai realizzatori di tool.
9. I documenti XML devono essere facili da creare.

   In particolare, deve essere facile creare tool di authoring di documenti XML.

Criteri di progettazione di XML (6)

10. Non ha importanza l'economicità del markup XML.
    - Le esigenze di economicità di markup (***terseness***) di SGML avevano portato all'adozione di molte pratiche di minimizzazione dei caratteri, che però rendevano i documenti poco leggibili e molto più complicati da parsare.
    - XML non ha meccanismi di minimizzazione, e dove si poteva scegliere tra economicità e chiarezza, si è scelta la chiarezza.

Esistono poi due obiettivi progettuali non riportati:

A. Supporto per l'internazionalizzazione
    - XML deve funzionare con tutti i set di caratteri.

B. Desperate Perl hacker
    - Il programmatore a cui viene imposto di eseguire un compito di modifica globale su una grande quantità di documenti e che riesce a farla applicando un qualche script semplice sulla struttura pulita dei documenti XML.

XML e Unicode

XML (come Java) abbandona completamente ASCII e le codifiche ad un byte, e si basa direttamente su Unicode.

Questo porta a due vantaggi nei riguardi dell'internazionalizzazione:

- È possibile scrivere documenti misti, senza ricorrere a trucchi strani per identificare la parte che usa un alfabeto dalla pare che ne adopera un altro.
- Un documento scritto in un linguaggio non latino non deve basarsi su parametri esterni per essere riconosciuto come tale, ma la codifica stessa dei caratteri lo identifica.

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
- I tag vuoti (senza contenuto) utilizzano un simbolo speciale di fine tag: <vuoto/>
- Tutti gli attributi sono sempre racchiusi tra virgolette
- Tutte le entità sono definite.

Parser validanti e non validanti

- Il cuore di un applicazione XML è il parser, ovvero quel modulo che legge il documento XML e ne crea una rappresentazione interna utile per successive elaborazioni (come la visualizzazione).
- Un parser validante, in presenza di un DTD, è in grado di verificare la validità del documento, o di segnalare gli errori di markup presenti.
- Un parser non validante invece, anche in presenza di un DTD è solo in grado di verificare la buona forma del documento.
- Un parser non validante è molto più semplice e veloce da scrivere, ma è in grado di fare meno controlli. In alcune applicazioni, però, non è necessario validare i documenti, solo verificare la loro buona forma.

Le novità sintattiche di XML

- La dichiarazione XML
- Sezioni CDATA
- Sintassi di <!ELEMENT>, degli elementi vuoti, delle processing instructions
- Rigore sintattico (case sensitivity, nessuna minimizzazione)
- Entità predefinite
- Gestione del white space
- Attributi riservati

Dichiarazione XML (1)

```
<?XML version="1.0" encoding="UTF-16" standalone="yes" ?>
```

- Un documento XML può includere una dichiarazione XML. Questa specifica le caratteristiche opzionali del documento in questione. Poiché esse sono ridotte al minimo, la dichiarazione XML è brevissima.
- La sintassi usata per la dichiarazione XML è quella delle Processing Instructions,
- La non obbligatorietà della dichiarazione XML è dovuta a motivi di convenienza, per poter usare la grande quantità di documenti HTML e SGML che sono ben formati senza richiedere modifiche anche stupide. In assenza di dichiarazione XML, si assume la forma:

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


Dichiarazione XML (2)

Esistono esattamente tre valori che possono essere messi in una dichiarazione XML:

- Il parametro “version” identifica quale versione di XML si sta usando. Per il momento, l’unico valore possibile è “1.0”. Necessario.
- Il parametro “encoding” permette di specificare, se il dubbio può sorgere, quale codifica di caratteri viene usata per il documento. Facoltativo.
- Il parametro “standalone” permette di specificare se le informazioni necessarie per valutare e validare il documento sono interne o se ne esistono anche di esterne. Facoltativo.

Sezioni CDATA

- A volte può essere comodo inserire un blocco di caratteri comprendenti anche ‘&’ e ‘<‘, senza preoccuparsi di nasconderli dentro ad entità.
- Si usa allora la sezione CDATA, che ha la seguente sintassi:

```
<![CDATA[ dati liberi comprendenti & e < ]]>
```

- L’unica sequenza di caratteri non accettabile è la sequenza ‘]]>’, che definisce la fine della sezione CDATA
- Il parser XML passa all’applicazione finale tutti i caratteri che trova fino alla sequenza]]>

```
<para>In HTML, “<![CDATA[ <h1>Questo &egrave; un
titolo</h1>]]>” indica un titolo </para>
```


Altre differenze tra SGML e XML (1)

- **Elementi vuoti**: un elemento con content model EMPTY ha il carattere di chiusura tag ‘/>’.

```
<EMPTY/>
```

- **Case sensitivity**: in XML tutto il markup è case-sensitive (il maiuscolo è diverso dal minuscolo). È quindi necessario usare le maiuscole per ELEMENT, ATTLIST, ecc., e l’elemento <para> è diverso dall’elemento <PARA>.
- **Valori tra virgolette**: tutti i valori di tutti gli attributi debbono avere le virgolette (semplici o doppie, ma in maniera coerente), anche se numeri o appartenenti ad una lista di valori predefiniti.

Altre differenze tra SGML e XML (2)

- **Tag omissibili**: Non esiste il concetto di tag omissibili, e nella definizione degli elementi non ci sono i parametri di minimizzazione.
- **Entità predefinite**: sono pre-definite e non ridefinibili 5 entità:

```
<!ENTITY lt     “&#60;”>
<!ENTITY gt     “&#62;”>
<!ENTITY amp    “&#38;”>
<!ENTITY apos   “’”>
<!ENTITY quot   ‘”’>
```

- **Processing instructions**: la sequenza di chiusura di un’istruzione di elaborazione è ‘?>’:

```
<?Fine-pagina?>
```


Il white space

XML adotta convenzioni molto semplici e dirette per il white space:

- ***New line***: Per semplicità ed uniformità, XML trasforma ogni tipo di new line (CRLF, LF e CR) nel solo carattere LF.
- "***If it ain't markup, it's data***": Ogni white space che appare nel contenuto del documento è rilevante, e deve essere passato intatto all'applicazione.
- Tuttavia, un ***parser validante*** è tenuto a precisare all'applicazione quale white space è stato riscontrato in elementi con content model di tipo elemento, cosicché l'applicazione possa decidere cosa farne.

Attributi per white space e lingua

Esistono in XML due attributi riservati (ma da definire se usati):

- **xml:space** (valori possibili: “default” o “preserve”) permette all’autore di indicare all’applicazione se è opportuno che mantenga il white space
- **xml:lang** (valori possibili: i codici a due lettere di RFC 1766): permette all’applicazione di identificare la lingua in cui è scritto il contenuto di un elemento, per attivare funzionalità dipendenti dalla lingua:
  - Rendering corretto
  - Spell-checking
  - Full-text indexing
  - Editing

Conclusioni

Qui abbiamo parlato di XML, soprattutto per quanto non è derivato da SGML:

- Il senso di XML
- Usi innovativi di XML
- I criteri progettuali
- La distinzione tra documenti ben formati e validi
- Le principali differenze sintattiche

Riferimenti

Wilde's WWW, capitolo 7

Altri testi:

- Neil Bradley, ***The XML companion***, Addison Wesley 1998
- J. Bosak, ***XML, Java, and the future of the Web***, `http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm`
- T. Bray, J. Paoli, C.M. Sperberg-McQueen, ***Extensible Markup Language (XML) 1.0***, W3C Recommendation, 10 February 1998, **`http://www.w3.org/TR/REC-xml`**
- T. Bray, The annotated XML Specification,1998, **`http://www.xml.com/axml/testaxml.htm`**
- XMLNews Specifications, **`http://www.xmlnews.org/`**