Perché XML?

Fabio Vitali

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

Documenti auto-descrittivi

- La scelta dei nomi degli elementi può essere fatta per facilitare la comprensione del ruolo strutturale dell'elemento.

Struttura navigabile dei documenti

- La rigida struttura ad albero e l'assenza di regole di minimizzazione rendono semplice la visualizzazione

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 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
- XSLT: un linguaggio di stylesheet per XML
- XML schema: un linguaggio per la specifica di criteri di validazione di documenti XML

Cosa si fa con XML?

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? (2)

Interazione tra 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? (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? (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? (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.

Quando scegliere XML? (1)

Quali sono le condizioni per adottare XML in un progetto?
Ovviamente:

- E' nuovo
- E' di moda
- E' compatibile con Web
- Può essere imposto dal committente
- Può essere imposto dai partner

Ma ci sono almeno quattro ***buone*** ragioni per XML:

- Produzione di documenti automatici
- Gestione indipendente di produzione e uso di dati
- Elaborazione di dati con aspetti strutturali complessi
- Elaborazione di dati strutturati in contenitori semi-strutturati

Quando scegliere XML? (2)

Produzione di documenti automatici

- XML è la soluzione in assoluto più elegante (anche se ad oggi ancora faticosa) per integrare collezioni di dati strutturati sul Web.
- Documenti dinamici, che mescolano blocchi testuali con output tabellari di informazioni strutturate, sono facilmente esprimibili in XML, e gli strumenti attuali si concentrano su questo, per il momento.
- Integra e sostituisce le tecnologie server-side di accesso ai dati: ASP, PHP, server-side Javascript, ecc.

Quando scegliere XML? (3)

Gestione indipendente di produzione ed uso di dati

- Spesso l'interscambio di dati avviene all'interno di un workflow controllato e noto. In questo caso, ***data producers*** e ***data consumers*** sono creati ad hoc per lo specifico flusso informativo. XML è una complicazione inutile.
- Tuttavia esistono delle situazioni in cui non c'è progettazione integrata di producer e consumer. In questo caso, un'adeguata progettazione del producer facilita molto il lavoro di tutti i possibili consumer
- XML è strutturato, auto-esplicativo, enfatizza la descrizione del dato più che del suo scopo nella elaborazione. E' quindi ideale per le situazioni in cui l'elaborazione non è nota in anticipo.

Quando scegliere XML? (4)

Elaborazione di dati con aspetti strutturali complessi

- I database utilizzano le relazioni per ogni tipo di esigenza: dalla descrizione di connessioni logiche tra entità concettualmente diverse, alla gestione di dati strutturati in maniera complessa.
- Ad esempio, è complicato gestire, in una tabella, record con un numero variabile di campi, o situazioni alternative complesse.
- XML prevede strutture con blocchi ripetuti, alternativi, facoltativi. La descrizione di queste strutture in XML è molto più ***naturale*** che con DB relazionali.

Quando scegliere XML? (5)

Elaborazione di dati in contenitori semi-strutturati

- A volte l'informazione ha uno stato naturale semi-strutturato (e.g., documenti testuali), al cui interno esistono informazioni atomiche su cui è necessario attivare computazioni.
- La soluzione classica è di estrarre le informazioni atomiche, metterle in un DB tradizionale, e buttare via il contenitore naturale. Questo ha il grosso difetto di eliminare il contesto e omogeneizzare in maniera forzata informazioni organizzate diversamente.
- XML permette di inserire all'interno di strutture documentarie (pensate per la visualizzazione) tag di natura descrittiva utilizzabili per elaborazioni sofisticate.

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>Un terremoto ha colpito la Colombia occidentale
         lunedì, uccidendo almeno 143 persone e ferendone
         più di 900 mentre scoperchiava edifici nella zona
         delle coltivazioni di caffè più ricca e fertile
         della nazione. Gli addetti alla difesa civile sono
         immediatamente intervenuti.</p>
      <p>Il terremoto è avvenuto nel primo pomeriggio, con
         una magnitudine del 6 grado, secondo il Geological
         Survey Americano, in Golden, Colorado. L'epicentro
         è stato identificato nella valle dello stato del
         Cauca, a 210 chilometri a ovest della capitale,
         Bogotà.</p>
</body.content> </body>
  </nitf>
```


Un esempio: XMLNews (4)

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

```
<p>Un <event>terremoto</event> ha colpito la <location>
   <country>Colombia</country> occidentale</location>
   <chron norm="19990125">Lunedì</chron>, uccidendo
   almeno 143 persone e ferendone più di 900 mentre
   scoperchiava edifici nella zona delle coltivazioni
   di caffè più ricca e fertile della nazione. Gli
   <function>addetti alla difesa civile</function>
   sono immediatamente intervenuti.</p>
```

Questo permette di arricchire la storia con altre informazioni 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.

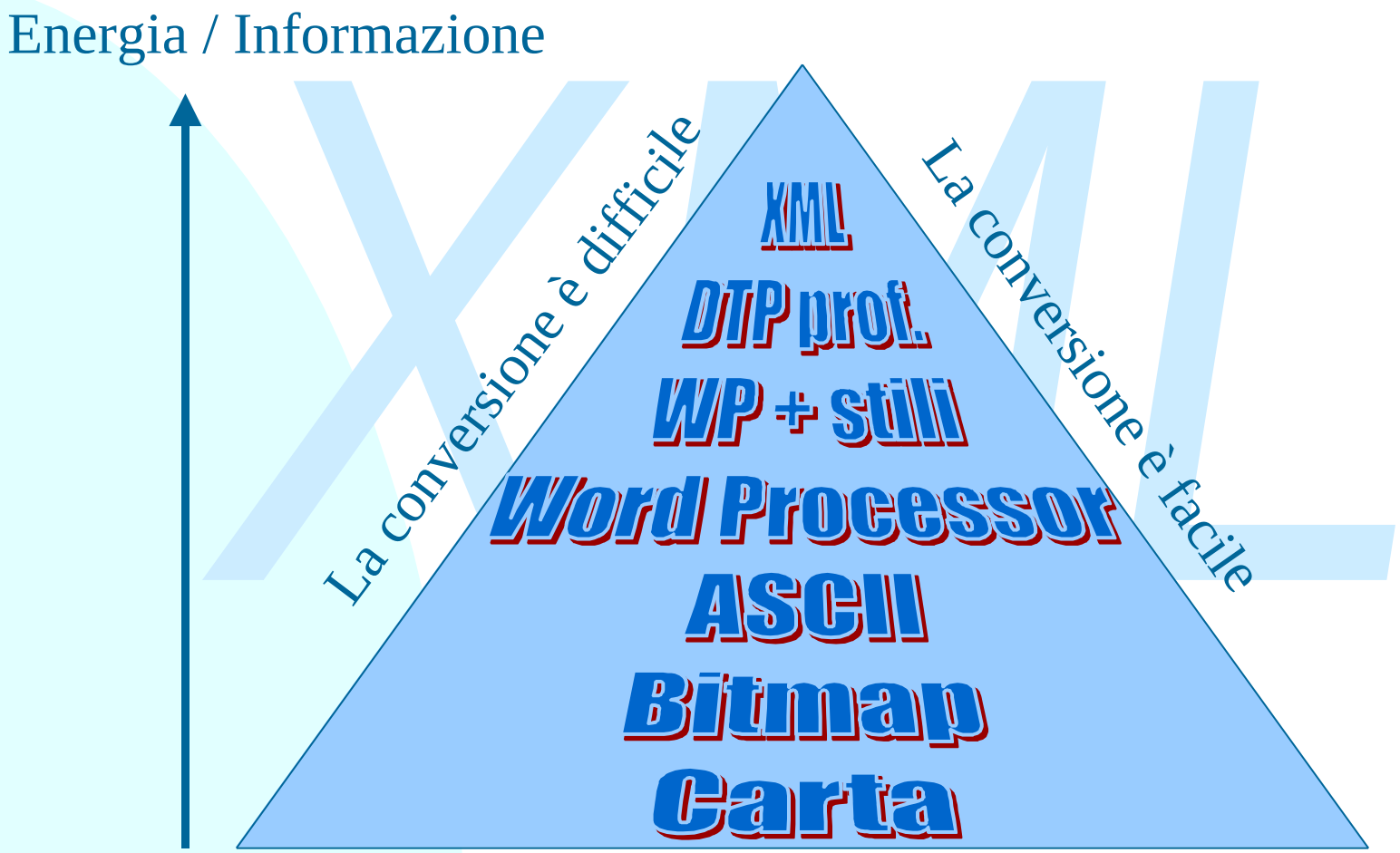
XMLNews-Meta permette di gestire insieme informazioni come:

- 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.

Analisi dei documenti

- Classificazione dei componenti
- Selezione dei componenti, costruzione della gerarchia, dei blocchi informativi e degli elementi di dati
- Identificazione delle connessioni
- Verifica e miglioramento iterativo delle specifiche

Perché convengono i linguaggi di markup?



Classificazione dei componenti

- La parte più importante del lavoro di progettazione di un'applicazione XML è l'identificazione delle strutture e del significato delle parti dei documenti e delle loro relazioni.
- L'identificazione semantica dei componenti avviene quando si evidenzia l'esigenza di distinguere tra un tipo di dati ed un altro.
- Se due pezzi diversi di un documento contengono lo stesso tipo di informazione, li si deve considerare appartenenti alla stesso componente semantico, anche se non sono contigui.
- Viceversa, se due pezzi contengono due tipi diversi di informazione, o è necessario distinguerli in qualche maniera per i fini dell'applicazione, allora debbono essere distinti in due componenti separati.
- Possiamo distinguere tra tre tipi di classificazione dei componenti di un documento: ***contenuto, struttura*** e ***presentazione***.

Classificazione basata sul contenuto

Si identificano i componenti per il significato che essi hanno, indipendentemente dalla loro posizione nel documento o dal loro aspetto grafico.

Ad esempio:

- indirizzi, città, codici postali;
- ricette, ingredienti, tempi di preparazione;
- termini, sviluppo grammaticale, significato.

Classificazione basata sulla struttura

Si identificano i componenti per il loro ruolo all'interno del documento, per il senso che hanno in quella posizione e in quella forma

Ad esempio:

- sezioni, capitoli, liste, paragrafi, titoli

Classificazione basata sulla presentazione

Si identificano i componenti per le variazioni nel modo in cui debbono apparire graficamente, senza implicazioni sul loro “vero significato”.

Ad esempio:

- Frasi con un determinato font o dimensione
- Blocchi da mantenere sulla stessa pagina
- Posti dove spezzare una pagina

Caratteristiche delle classificazioni

Questi tre modi di classificare i componenti di un documento sono presenti contemporaneamente bell'analisi di un documento. Persone diverse, sugli stessi documenti, possono identificare questa o quella classe a seconda di professione, ***forma mentis***, esigenze.

I tre tipi di classificazione hanno anche caratteristiche diverse di ***identificabilità, flessibilità*** e ***durata***.

Classificazione basata sul contenuto

È la classificazione più complessa da realizzare, ma la più flessibile, identificabile, flessibile e duratura.

Poiché identifico i componenti basati sul loro significato, è immediato arrivare al senso di un componente.

La classificazione è indipendente dalla struttura del documento e dall'aspetto grafico, così posso cambiare idea su queste decisioni in qualunque momento, e anche fornire soluzioni diverse sugli stessi componenti.

Poiché un componente avrà sempre quel significato in qualunque contesto, modifiche importanti, come cambiamenti di stile, organizzazione del documento ecc. non impediranno a questa classificazione di sopravvivere.

Classificazione basata sulla struttura

La classificazione strutturale identifica l'organizzazione di un documento in maniera sufficientemente generale, ma rozza per quel che riguarda il senso del documento.

È possibile in qualunque momento modificare la resa grafica degli elementi, ma non il loro ruolo nella struttura globale del documento.

Cambiamenti stilistici non preoccupano (il font, la larghezza di un paragrafo, l'esistenza o meno di un bordo in una tabella), cambiamenti strutturali importanti invece sì (ad esempio, passare da una forma a lista ad una a tabella, trasferire contenuto da una sezione all'altra, ecc.).

Classificazione basata sulla presentazione

In generale, ci sono più classi di contenuto che classi di presentazione; questo significa che, per uniformità grafica e facilità di lettura, molti componenti con semantica diversa vengono resi graficamente nello stesso modo.

La specificazione delle sole classi grafiche fa sparire immediatamente l'identificabilità di elementi di significato diverso ma resa grafica uguale.

Decisione successive di cambiamenti grafici di solo alcuni componenti, e non altri, saranno impossibili.

Utilizzi del documento per scopi diversi dalla presentazione (ad esempio, la creazione di un indice, l'inserimento in un motore di ricerca, ecc.), saranno impossibili.

Regole guida per la classificazione

1. Identificare il più possibile i componenti per il loro significato e contenuto. Richiede più lavoro ma ne vale la pena. È necessario, ovviamente, fermarsi ad un livello ragionevole di specificità.
2. Attribuire a questi componenti significati strutturali (tabelle, liste, paragrafi, organizzazioni gerarchiche tipo sezioni, sotto-sezioni, ecc.).
3. Specificare la resa grafica dei componenti. Quest'ultima specificazione tipicamente avviene al di fuori del contesto di XML, con appositi strumenti e linguaggi di stylesheet.

Altri suggerimenti (1)

Scartare elementi esclusivamente presentazionali: numeri di pagina, elementi ripetuti (un logo, una decorazione, il nome di un capitolo), ecc. Essi possono essere aggiunti automaticamente dal formattatore e non è necessario considerarli come componenti del documento.

Identificare classi generali di informazioni. Anche se presenti in varie parti del documento, alcune informazioni possono avere lo stesso significato e lo stesso ruolo, e quindi debbono essere identificati nella stessa maniera.

Altri suggerimenti (2)

Identificare informazioni ripetute in varie parti del documento. Alcune informazioni (nomi propri o di organizzazioni, riferimenti ad immagini, date importanti, elementi ripetuti di una struttura, ecc.) debbono essere presenti in varie parti del testo in maniera identica, e debbono cambiare in maniera coerente.È utile avere un componente unico che registri una sola volta l'informazione da stampare, e venga usato ovunque necessario.

Identificare i componenti che provengono da sistemi informativi esistenti. Tipicamente un database ha già distinzioni di elementi basate sul contenuto. Se alcune informazioni provengono da un database è comodo e si risparmia tempo usare o basarsi sulla strutturazione dei dati già esistenti nel sistema informativo.

Conclusioni

Qui abbiamo parlato delle motivazioni per usare un linguaggio di markup (ed in particolare XML):

- La giustificazione di XML
- Le ragioni per scegliere XML
- L'analisi dei documenti
- Le regole di classificazione

Riferimenti

Wilde's WWW, capitolo 4

Altri testi:

- E. Maler, J. Al Andaloussi, ***Developing SGML DTDs, from text to model to markup***, Prentice Hall, 1997
- N. Bradley, ***The XML companion***, Addison-Wesley, 1998
- C.F. Goldfarb, ***The SGML handbook***, Clarendon Press, 1990