XML e standard connessi

XML, XML-Names, XSL, XPointer, XLink

Sommario

- Introduzione
- XML 1.0
- XML-Names
- XSL
- XPointer e XLink

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.

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 di:
  - **XML 1.0: un meta-linguaggio di markup, sottoinsieme di SGML**
  - **XML-Names: un meccanismo per la convivenza di nomi di tag appartenenti a DTD diversi**
  - **XSL: un linguaggio di stylesheet per XML**
  - **XPointer e XLink: due linguaggi connessi per la creazione di link ipertestuali**

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 ufficiosa in italiano: http://www.iat.cnr.it/xml/REC-xml-19980210-it.html
- Molto più formalizzata della grammatica di SGML, usa la notazione ***EBNF*** (***Extended Backus-Naur Form***), un formalismo ben noto in tutti gli ambienti di programmazione (usata normalmente per descrivere la grammatica dei linguaggi di programmazione).

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 semplice su Internet.**
  2. **XML deve supportare un gran numero di applicazioni.**
  3. **XML deve essere compatibile con SGML.**
  4. **Deve essere facile lo sviluppo di programmi che elaborino documenti XML.**
  5. **Il numero di caratteristiche opzionali deve essere mantenuto al minimo possibile, idealmente a zero.**

Criteri di progettazione di XML: 2

6. **I documenti XML dovrebbero essere leggibili da umani e ragionevolmente chiari.**
7. **La progettazione XML dovrebbe essere rapida.**
8. **La progettazione XML deve essere formale e concisa.**
9. **I documenti XML devono essere facili da creare.**
10. **Non è di nessuna importanza l'economicità nel markup XML.**

ASCII e ISO-Latin 1

- Il codice ASCII è stato inventato per associare lettere dell'alfabeto inglese ai codici numerici usati internamente dai computer.
- Il codice ASCII originale usava le prime 128 combinazioni di un byte (0-127), lasciando il bit più significativo come controllore di parità.
- In seguito, la superfluità del codice di parità ha portato estensioni non-standard del codice ASCII per i rimanenti 128 caratteri (128-255).
- ISO-Latin 1 è uno standard ISO per i caratteri derivati dall'alfabeto latino, che occupa tutti i 256 caratteri del byte. I primi 128 sono lo standard ASCII.

Unicode

- Unicode è una codifica a due byte che include ISO-Latin 1 (i primi 256 caratteri sono ISO-Latin 1) e permette di specificare tutti i caratteri di una moltitudine di alfabeti, includendo: Greco, Copto, Cirillico, Armeno, Ebraico, Arabo, Hiragana e Katakana (i due principali alfabeti giapponesi), e Han (alfabeto ieratico di cinese, giapponese e coreano), ecc.
- Inoltre è a disposizione uno spazio libero da utilizzare per tutti gli alfabeti non inclusi (giapponese e cinese arcaico, cuneiforme, geroglifico, Klingon, ecc.)

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.

Dichiarazione XML

- Un documento XML può includere una dichiarazione XML. Poiché le caratteristiche opzionali di XML sono ridotte al minimo, la dichiarazione XML è brevissima:
  <?XML version=“1.0” encoding=“UTF-16” standalone=“yes” ?>
- 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 accettata è la sequenza ‘]]>’.
- In una sezione CDATA il parser XML passa all’applicazione finale tutti i caratteri che trova fino alla sequenza]]>

Altre macro-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 macro-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 da non ridefinire 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?>
```

XML-Names

- Nella visione XML, i DTD si mescolano e si fondono tra loro in maniera complessa. Lo stesso documento potrebbe avere alcuni elementi definiti in un DTD ed altri in un altro.
- Un esempio comune è un documento XML di valori di borsa che adopera il DTD di HTML per definire gli elementi di testo, ed un DTD specifico per gli elementi di borsa.
- I problemi sono conciliare la presenza di elementi non definiti nel content model e soprattutto conciliare la presenza di elementi definiti nei due DTD con lo stesso nome.
- I ***namespace*** in XML si propongono per risolvere questi problemi. XML-names è un ***working draft*** di W3C per questo argomento.

Un esempio di uso di XML-Names

- Il seguente esempio usa i ***namespace*** XML:

```
<?xml version="1.0"?>
<bk:book xmlns:bk="http://www.loc.gov/books"
         xmlns:isbn="ISBN">
    <bk:title>Tre uomini in barca</bk:title>
    <bk:author>Jerome K. Jerome</bk:author>
    <isbn:isbn>88-7983-395-2</isbn:isbn>
</bk:book>
```

- Ogni nome del documento XML è preceduto da un prefisso che specifica l'origine del nome stesso. Questo può avvenire sia per i nomi degli elementi che per i nomi degli attributi. Il prefisso è separato da il carattere ':' dal nome dell'elemento o dell'attributo.
- Il nome predefinito "xmlns" serve per introdurre i namespace usati nel documento e per fornire un identificatore pubblico dove trovare i dettagli del namespace (ad esempio, il DTD).

Namespace di default

- Nella dichiarazione ***xmlns*** si pone il nome del prefisso che si intende usare nel corso del documento per gli elementi definiti in quel ***namespace***.
- L'assenza di tale prefisso in xmlns indica la presenza di un namespace di default, per cui tutti i nomi privi di prefisso di debbono intendere appartenenti a quel namespace.
- Es.:

```
<?xml version="1.0">
<book xmlns="http://www.loc.gov/books"
          xmlns:isbn="ISBN">
    <title>Tre uomini in barca</title>
    <author>Jerome K. Jerome</author>
    <isbn:isbn>88-7983-395-2</isbn:isbn>
</book>
```

XPointer e XLink

- XLink e XPointer sono due ***working draft*** di W3C per la specifica di link ipertestuali sui documenti XML.
- Originariamente erano un'unica proposta chiamata XLL (da cui la terna XML, XLL e XSL), poi divisa in due per semplicità.
- XPointer specifica i meccanismi per riferirsi a parti del documento XML (SGML permette di riferirsi solo ad elementi con l'attributo "ID", HTML solo ad elementi con l'attributo "NAME").
- XLink usa i meccanismi di indirizzamento di XPointer per descrivere link anche sofisticati tra documenti XML.

XPointer

- Gli XPointer sono indirizzi di locazioni interne a documenti XML. Gli XPointer possono essere usati per indicare link da o a specifiche parti di documenti XML.
- Gli XPointer sono una elaborazione del nome in un URL: `http://www.domain.com/dir/file.html#`***`nome`***
- Gli XPointer sono dunque usati in un locatore, tipicamente un URI o URL, per indicare un frammento di quella risorsa.
- Un XPointer è composto di una sequenza di termini di locazione separati da punti. Ogni termine individua più precisamente un frammento della risorsa individuata in precedenza.
- Es.: **`root().child(2,DIV)`** identifica il secondo elemento “DIV” figlio diretto della radice del documento XML.

Tipi di indirizzamento: 1

I termini di locazione possono essere:

- **Assoluti**: elementi identificabili senza ambiguità, come la radice di un albero, il punto di partenza della navigazione, un elemento dotato di un attributo ID, un elemento HTML dotato di un attributo “NAME”.
  - **Es.: `root(), origin(), id(“pippo”), html(“MyName”)`**
- **Relativi**: elementi basati sul termine precedente del XPointer, e identificati tramite relazioni di vicinanza e di tipo: n-esimo figlio, discendente, ascendente, precedente, seguente, ecc.
  - **Es.: `child(2,SECTION), preceding(4,DIV), descendant(4,#text)`**

Tipi di indirizzamento: 2

- **Intervallo (span)**: un frammento in cui inizio e fine sono indicati da due ulteriori frammenti di XPointer.
  - **Es.:** `id(27).span(child(1), child(5))`
- **Attributi**: un frammento indicato dalla presenza o meno di un attributo.
  - **Es.:** `id(27).attr("pos")`
- **Stringhe**: il frammento identificato da o una posizione, o una stringa specifica.
  - **Es.:** `string(15,"")` **identifica il carattere n. 15 della stringa precedentemente identificata.**
  - **Es.:** `string(3,"text",5)` **identifica il carattere n.5 della occorrenza n. 3 della stringa "text" nella stringa precedentemente identificata.**

XLink

- Gli XLink sono elementi di un documento XML che hanno significato e comportamento di link ipertestuale.
- Un elemento XML è identificato come un XLink se possiede degli attributi riservati, come **`xml:link`**.
- Questa è una soluzione di compromesso tra il riservare nomi specifici di elementi (che avrebbe limitato la libertà di creazione dei DTD) e il lasciare tutta la gestione dei link ai fogli di stile (che avrebbe negato di attribuire inequivocabilmente agli elementi in questione la natura di link).
- Gli XLink sono di due tipi:
  - **Link semplici: elementi inline e uni-direzionali**
  - **Link estesi: inline o out-of-line, spesso multi-direzionali.**

XLink semplici

- La presenza dell'attributo **`xml:link`** con valore "simple" identifica l'elemento come un XLink semplice.

```
<A xml:link="simple" href="http://www.w3.org/">W3C</A>
```

- Un modo più semplice di definire elementi è specificare in un DTD (o anche in un frammento inline di DTD) la presenza dell'attributo:

```
<!ATTLIST A  xml:link  CDATA #IMPLIED "simple">
<A href="http://www.w3.org/">W3C</A>
```

- L'attributo **`href`** identifica il o i locatori coinvolti. Un locatore è un URL seguito facoltativamente da un XPointer.

```
<A href="http://www.w3.org/file.xml#root().child(3,DIV)">W3C</A>
```

XLink estesi

- La presenza dell'attributo **`xml:link`** con valore "extended" identifica l'elemento come un XLink esteso.

```
<commenti xml:link="extended" inline="false">
  <nota xml:link="locator" href="#id(n1)" role="supporto"/>
  <nota xml:link="locator" href="#id(n2)" role="critica"/>
</commenti>
```

- Un link esteso può avere più di un locatore. In questo caso si usano elementi con attributo **`xml:link`** = "locator".
- I link estesi sono utili per:
  - **Creare link in uscita da documenti che non possono essere modificati (e quindi non permettono link inline)**
  - **Creare link da o per documenti in formati non XML.**
  - **Eseguire l'attivazione di collezioni di link a richiesta.**
  - **Specificare link mutli-direzionali e multi-destinazione.**

Link inline e out-of-line

- Un link HTML è un link inline: il testo linkante o linkato è il contenuto dell'elemento A, e il link appartiene al documento in cui appare.
- I link out-of-line sono link che vengono memorizzati in un documento, e appaiono in un altro. Questo è molto comodo per creare link a o da risorse read-only (CD-ROM) o su cui non si hanno permessi di modifica (documenti altrui).
- XLink determina il tipo di link tramite l'attributo **`inline`**, che può avere valori "true" o "false".

Semantica della risorsa locale

- La risorsa locale, in un link inline, è la parte di documento contenuta nell'elemento XLink. In link non inline, non esiste risorsa locale.
- Con XLink è possibile specificare degli attributi per arricchire l'interpretazione della risorsa locale. Essi sono:
  - **`Content-role`: specifica il ruolo che gioca la risorsa locale nel link. È un meccanismo di tipizzazione.**
  - **`Content-title`: specifica un titolo visualizzabile per spiegare all'utente il ruolo del link, la destinazione, ecc.**

Semantica della risorsa remota: 1

- La risorsa remota è l'elemento a cui punta un locatore. Nel caso di link unidirezionale inline, la risorsa remota è la destinazione del link. In altri casi, è semplicemente uno degli estremi.
- Con XLink è possibile specificare degli attributi per arricchire l'interpretazione della risorsa remota. Essi sono:
  - **`role`: specifica il ruolo che gioca la risorsa remota nel link. È un meccanismo di tipizzazione.**
  - **`title`: specifica un titolo visualizzabile per spiegare all'utente il ruolo del link, la destinazione, ecc.**
  - **`behavior`: specifica dei comportamenti che l'applicazione deve attuare durante l'attravrsamento (effetti visivi, sonori, ecc.)**

Semantica della risorsa remota: 2

- Altri due importanti attributi sono:
  - **show (valori possibili: “embed”, “replace”, “new”): specifica come visualizzare o elaborare la risorsa specificata. “new” indica che la risorsa va visualizzata in un contesto nuovo, come una finestra nuova; “replace” indica che la risorsa nuova sostituisce la vecchia nel contesto esistente. “embed” indica che il contesto della risorsa nuova è la risorsa locale, a cui va sostituita. Per esempio, rimpiazzando il testo del link con il testo della destinazione.**
  - **Actuate (valori possibili: “auto” e “user”): specifica quando l’attivazione del link debba avvenire. “user” indica che deve essere l’utente ad attivare l’azione, per esempio facendo click su un pulsante. “auto” significa che il link deve essere attivato appena la risorsa locale viene caricata.**

Gruppi di link estesi

- Grazie ai link estesi, è possibile mettere in due documenti diversi i link ed il testo che li deve contenere.
- Nasce allora l'esigenza di specificare un gruppo di documenti connessi tra loro per l'esistenza di link incrociati, cosa che avviene con una derivazione della stessa idea di link esteso.
- La presenza dell'attributo **`xml:link`** con valore "group" identifica l'elemento come un gruppo di link estesi. La presenza dell'attributo **`xml:link`** con valore "document" identifica l'elemento come un documento interessato dal gruppo..

```
<gruppo xml:link="group">
  <doc xml:link="document" href="doc.xml" role="text"/>
  <doc xml:link="document" href="extlink1.xml" role="link"/>
  <doc xml:link="document" href="extlink2.xml" role="link"/>
</gruppo>
```

XSL: Un linguaggio di stylesheet

- Poiché nessun elemento di XML possiede un significato predefinito, il linguaggio di stylesheet si occupa di dare un ***significato*** agli elementi di un documento XML.
- XSL (Extended Stylesheet Language) è un linguaggio che permette di attribuire significati “ben noti” (come caratteri, font, ecc.) agli elementi di un documento XML.
- Il linguaggio XSL non ha ancora uno stato concluso. E’ definito da un working draft di aprile 1999, che avrà ancora delle variazioni prevedibili.
- La proposta è divisa in due parti: un linguaggio di trasformazione da documenti XML a documenti XML, ed un vocabolario di elementi XML con semantica di formattazione.

Come funziona XSL

- Innanzitutto XSL è un linguaggio di trasformazione: dato un documento XML, è possibile generare un altro documento XML derivato applicando delle regole di trasformazione specificate nello stylesheet.
- Se il documento XML derivato contiene elementi i cui nomi ed attributi sono noti ad un browser, il documento può essere visualizzato da questo browser.
- Per esempio, nel caso di Internet Explorer 5.0, il documento XML viene trasformato dallo stylesheet XSL in un documento HTML (o meglio, XML con il DTD HTML), che può quindi essere visualizzato all'interno di un browser WWW.
- Un browser completamente compatibile con XSL, però, realizza anche il vocabolario di elementi di formattazione (***formatting objects***) che specificano sofisticate caratteristiche di impaginazione.

Cos’è un documento XSL?

- Un documento XSL è un documento XML che utilizza un DTD speciale in cui gli elementi hanno senso predefinito.
- Un documento XSL è composto di ***regole di costruzione***, le quali permettono di associare certi elementi del documento XML d’origine in altri elementi del documento destinazione.
- Una regola è composta di uno o più ***pattern*** (serie di criteri da verificare nel documento d’origine) e da una ***azione*** (sequenza di costrutti XML da inserire nel documento destinazione).
- Un parser XSL valuta, per ogni elemento del documento d’origine, la regola più appropriata da attivare, ed esegue l’azione associata, generando nuovi elementi XML al posto dei vecchi.

Un esempio

Dato lo stylesheet:

```
<xsl:stylesheet>
    <xsl:template match="/">
        <HTML><BODY>
            <xsl:process-children/>
        </BODY></HTML>
    </xsl:template>
    <xsl:template match="para">
        <P color="blue">
            <xsl:process-children/>
        </P>
</rule></xsl>
```

ed il documento

```
<doc>
    <para> Prova </para>
    <para> XSL </para>
</doc>
```

ne risulta il documento:

```
<HTML><BODY>
  <P color="blue"> Prova </P>
  <P color="blue"> XSL </P>
</BODY></HTML>
```

Pattern: l’attributo “match”

- Un pattern è un blocco di regole di costruzione contenute nell’attributo ***match***.
- Il pattern è l’elemento del documento sorgente a cui applicare l’azione. La regola fa match con tutti gli elementi che soddisfano le proprietà del pattern.
- Es.: La regola

```
<xsl:template>
      <DIV>
            <xsl:process-children/>
      </DIV>
</xsl:template>
```

  fa match con tutti gli elementi del documento.

Azione: gli elementi destinazione

- Sono un’azione del pattern tutti gli elementi della regola contenuti dentro al blocco <xsl:template>.
- Ogni elemento posto nell’azione viene semplicemente inserito nel documento destinazione al posto dell’elemento che ha fatto match.
- Fanno eccezione gli elementi speciali come <xsl:process-children> e in generale tutti quelli appartenenti al namespace “xsl:”.

Azione: <xsl:process-children/>

- L’elemento <xsl:process-children/> attiva la procedura di match sul #PCDATA e sugli elementi di markup contenuti nel elemento prescelto. L’elemento <xsl:process-children/> può comparire anche più volte.
- Es.: La seguente regola non scrive il contenuto dell’elemento “hide”.

```
<xsl:template match=“hide”/>
</xsl:template>
```

- Es.: La seguente regola scrive due volte il contenuto dell’elemento ripeti separandolo con una riga.

```
<xsl:template match==“ripeti”/>
      <DIV>
            <xsl:process-children/>
            <HR/>
            <xsl:process-children/>
        </DIV>
</xsl:template>
```

Usare XSL

- E’ necessario indicare al browser dove trovare lo stylesheet da usare. Questo può essere fatto in tre modi:
  - Specificando nell’intestazione MIME del collegamento HTTP la locazione del foglio di stile
  - Specificando un gruppo di documenti, uno dei quali è il foglio di stile
  - Specificando con una Processing Instruction (PI) il collegamento:

```
<?xml-stylesheet type=“text/xsl” href=“style.xsl”>
<doc> … </doc>
```

- L’ultima soluzione è l’unica che funziona oggi con Internet Explorer 5.0, ma non è ancora ufficiale. In prospettiva, credo che la soluzione migliore sia la seconda.