

Xml e xsl: tutorial introduttivo

Marco Deseri

Scopriamo xml per gradi: cos'è e perché può esserci utile, come scrivere codice xml e come realizzare pagine html grazie a xsl.

Cos'è Xml

Xml è un linguaggio di markup flessibile, adatto a descrivere la struttura e la semantica di un documento.

Come html, xml è un linguaggio di markup: testo delimitato da marcatori, detti tag. Xml non ha marcatori predefiniti, a differenza di html, perché è un linguaggio più astratto.

Html è nato per descrivere documenti informativi e dispone di una semantica adatta allo scopo: all'interno del documento vengono identificate porzioni di testo che hanno il valore di titolo di primo livello (h1), di paragrafo (p) e così via.

Xml, invece, consente di definire semantiche diverse in base al tipo di informazione da descrivere, che può essere di qualsiasi natura. Si può usare Xml per descrivere un album musicale, una pianta, il campionato di calcio. Per esempio, la descrizione di un album musicale potrebbe prevedere: titolo, nome dell'artista, nome del produttore ed elenco delle canzoni. Con xml è possibile costruire un linguaggio di markup adatto allo scopo.

Vantaggi

I principali vantaggi di xml? Eccoli.

- i documenti xml sono adatti ad essere trattati dalle macchine: gli elaboratori possono percorrere la struttura di un file xml a caccia di un'informazione;
- i documenti xml sono portabili: poiché rispettano una struttura rigida, si prestano ad essere trasformati con procedure automatiche, definite in fogli di stile appositi;
- i documenti xml sono riusabili: la struttura è chiaramente definita e consente di reperire aggregati di informazioni anche inediti.

Xml dà un senso al contenuto: non è un formato pensato per la visualizzazione da parte degli esseri umani, anzi, molto spesso xml è usato come formato di interscambio tra macchine.

Tuttavia, xml è molto adatto anche per gestire contenuti destinati alle persone, per via della sua trasformabilità. I contenuti xml possono essere convertiti tramite fogli di stile in formati di presentazione, come html o pdf. Sul tema della gestione dei contenuti con xml, potete vedere l'articolo [Contenuti portabili con Docbook](http://www.i-use.it/approfondimenti/contenuti/2/) (<http://www.i-use.it/approfondimenti/contenuti/2/>).

Un po' di codice xml

Xml consente una grande flessibilità sul piano semantico, ma richiede una sintassi formalmente corretta. Per essere considerato *ben formato* un file xml deve rispettare queste regole:

- ci deve essere un elemento radice, e deve essere unico;

- gli elementi e i nomi degli attributi devono essere scritti in minuscolo;
- per gli elementi non vuoti è richiesto il tag di chiusura;
- i valori degli attributi vanno sempre racchiusi tra virgolette doppie;
- anche gli elementi vuoti devono essere chiusi, oppure devono contenere uno slash "/" in fondo;
- gli elementi devono essere annidati correttamente;

E adesso vediamo un semplice file xml di esempio (canzoni.xml) (salvate il file come canzoni.xml):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<album>
  <titolo>Hail to the thief</titolo>
  <artista>Radiohead</artista>
  <produttore>Nigel Godrich</produttore>
  <anno>2003</anno>
  <etichetta>Capitol Records</etichetta>

  <canzone>
    <numero>1</numero>
    <titolo>2+2=5</titolo>
  </canzone>

  <canzone>
    <numero>2</numero>
    <titolo>Sit down, stand up</titolo>
  </canzone>

    <canzone>
      <numero>3</numero>
      <titolo>Sail to the moon</titolo>
    </canzone>

  <canzone>
    <numero>4</numero>
    <titolo>Backdrifts</titolo>
  </canzone>

  <canzone>
    <numero>5</numero>
    <titolo>Go to sleep</titolo>
  </canzone>

  <canzone>
    <numero>6</numero>
    <titolo>Where I end and you begin</titolo>
  </canzone>

  <canzone>
    <numero>7</numero>
    <titolo>We suck young blood</titolo>
  </canzone>

    <canzone>
      <numero>8</numero>
      <titolo>The gloaming</titolo>
    </canzone>
```

```

<canzone>
  <numero>9</numero>
  <titolo>There there</titolo>
</canzone>

<canzone>
  <numero>10</numero>
  <titolo>I will</titolo>
</canzone>

<canzone>
  <numero>11</numero>
  <titolo>A punch-up at the wedding</titolo>
</canzone>

<canzone>
  <numero>12</numero>
  <titolo>Myxomatosis</titolo>
</canzone>

  <canzone>
    <numero>13</numero>
    <titolo>Scatterbrain</titolo>
  </canzone>

<canzone>
  <numero>14</numero>
  <titolo>A wolf at the door</titolo>
</canzone>

</album>

```

Questo documento rispetta le regole che abbiamo visto, ed è un documento ben formato. Potete provare a fare un copia-incolla del codice in un qualsiasi editor di testo, e a salvare il file come canzoni.xml.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
- <album>
  <titolo>Hail to the thief</titolo>
  <artista>Radiohead</artista>
  <produttore>Nigel Godrich</produttore>
  <anno>2003</anno>
  <etichetta>Capitol Records</etichetta>
- <canzone>
  <numero>1</numero>
  <titolo>2+2=5</titolo>
</canzone>
- <canzone>
  <numero>2</numero>
  <titolo>Sit down, stand up</titolo>
</canzone>
- <canzone>
  <numero>3</numero>
  <titolo>Sail to the moon</titolo>
</canzone>

```

Aprendolo con un browser (da explorer 5 in avanti), dovrete visualizzare il documento come albero gerarchico.

E' molto importante avere in mente la struttura gerarchica del documento, perché è alla base di tutte le operazioni sul file che possiamo compiere, tra cui la trasformazione tramite xsl.

Per chi ha familiarità con dos o con una shell unix, l'albero del documento può essere percorso esattamente come un filesystem, usando questa notazione:

- “/” rappresenta un nodo; se è usato da solo rappresenta l'elemento radice;
- “.” è il livello di profondità a cui ci si trova al momento;
- “..” è il livello di profondità gerarchicamente superiore all'attuale;

Realizzare documenti html con xml

Per realizzare un documento html, a partire dal nostro canzoni.xml, è necessaria una trasformazione attraverso fogli di stile xsl.

I fogli di stile xsl

Extensible Stylesheet Language (xsl) comprende sia un linguaggio di trasformazione, sia un linguaggio di formattazione. Attraverso il linguaggio di trasformazione (xslt), di cui ci occuperemo, è possibile definire le regole con cui un documento xml viene trasformato in un altro documento xml.

La trasformazione del documento viene realizzata da un *parser xsl*, che:

- prende in input un documento xml e un foglio di stile xsl;
- applica le regole di trasformazione del foglio di stile;
- restituisce in output un documento xml ben formato (per esempio, un file xhtml);

Esistono molti parser xsl, generalmente gratuiti, per tutti i sistemi operativi. Internet Explorer dispone di un proprio parser, in grado di trasformare automaticamente un file xml attraverso xsl. Questo significa che, se all'interno del documento xml è linkato un foglio di stile, Explorer eseguirà automaticamente la conversione.

Per linkare un foglio di stile è sufficiente inserire all'interno del documento xml, subito dopo il prologo, una linea di codice simile a questa:

```
<?xml-stylesheet type="text/xsl" href="canzoni.xsl"?>
```

Linkare il foglio di stile direttamente al file xml e far eseguire la trasformazione al browser può essere una soluzione rapida per fare delle prove, ma non è certo l'ideale in ambiente di produzione.

Per creare documenti leggibili da tutti i browser è consigliabile pre-parsare i file xml e far servire dal proprio web-server semplici file html.

Dopo questa divagazione sui parser, torniamo ai fogli stile xsl. Il foglio di stile è costituito da *regole di matching* e da *template*: quando il parser incontra nell'albero del documento xml una corrispondenza con l'elemento da cercare, procede alla sostituzione con ciò che incontra nel template.

Vediamo un file xsl di esempio (canzoni.xsl) (salvate il file come canzoni.xsl, nella stessa directory di canzoni.xml):

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h1><xsl:value-of select="/album/titolo"/>
-
<xsl:value-of select="/album/anno"/></h1>
```

```

<h2><xsl:value-of select="/album/artista"/></h2>
<h3>Prodotto da:<xsl:value-of select="/album/produttore"/></h3>
<h3>Lista delle canzoni:</h3>

<table cellpadding="3" cellspacing="0" border="1">
<thead>
<tr>
<th>nr.</th><th>titolo</th>
</tr>
</thead>
<xsl:for-each select="/album/canzone">
<tr>
<td><xsl:value-of select="numero"/></td>
<td><xsl:value-of select="titolo"/></td>
</tr>
</xsl:for-each>
</table>

</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Le regole di sostituzione cominciano dalla terza riga:

```
<xsl:template match="/">
```

e terminano con:

```
</xsl:template>
```

Tra questi due punti, sono definite le regole da applicare quando il parser incontra l'elemento radice ("/").

Il contenuto di questo blocco è un misto tra html e regole di sostituzione. L'html viene mandato in output così com'è, mentre per le altre regole xsl il parser effettua le sostituzioni man mano che le incontra.

Simuliamo il comportamento del parser e vediamo in concreto cosa succede:

- Appena il parser incontra l'elemento radice del documento (nel nostro caso, album) stampa il codice html che incontra, senza modifiche: <html> <body> <h1>.
- Il parser incontra una regola di sostituzione (xsl:value-of select="/album/titolo") e stampa il valore del nodo /album/titolo. Poi, incontra un trattino "-" e lo stampa; incontra una regola ed esegue la sostituzione, stampando il valore del nodo /album/anno.¹.
- Il parser stampa il codice html che segue: </h1><h2>
- il parser incontra una nuova regola di sostituzione: come prima, percorre l'albero del documento a partire dall'elemento radice ed effettua la sostituzione per il nodo /album/artista. Poi incontra del codice html: la chiusura del tag h2 e l'apertura di una nuova tabella. Il parser incontra la regola <xsl:for-each select="/album/canzone">: questa è una regola ricorsiva, che viene applicata una volta per ogni ramo /album/canzone. Il template di sostituzione fa stampare, per ogni nodo, i tag <tr><td>, il valore dell'attuale nodo numero, il valore dell'attuale nodo titolo, il tag </td> e il tag </tr>.
- Infine, il parser stampa </table></body></html>.

Come vedete, il meccanismo è piuttosto semplice. Xsl è un linguaggio molto potente e consente di fare molto più di quello che abbiamo illustrato in questo banale esempio. In futuro, vedremo di analizzare più nel dettaglio le potenzialità di xml e xsl.

Risorse

Una guida sistematica, e adatta per cominciare, sull'argomento xml è quella di Html.it:

- <http://www.html.it/xml/guida/>

Non possono mancare i riferimenti alle pagine del W3C su xml e xsl

- <http://www.w3.org/XML/>
- <http://www.w3.org/Style/XSL/>

Note

1. Da notare ci sono due cose: la prima è che in questo caso il percorso è espresso in termini assoluti, partendo dall'elemento radice. La seconda è che il tag `<xsl:value-of> <liter>` è chiuso nella forma abbreviata, usando uno slash `<xsl:value-of />`, piuttosto che con un tag di chiusura separato `<xsl:value-of></xsl:value-of>`