


Linguaggi ed Applicazioni multimediali

06.05 XLST

Maurizio Maffi

ISTI Information Science and Technology Institute


Da XSL a XSLT

XSL o *eXtensible Stylesheet Language*, è un linguaggio XML creato inizialmente per **fornire a XML un supporto per la formattazione simile a CSS**, ma potenziato per riflettere il linguaggio su cui viene applicato, che non è più HTML

XSLT (*XSL Transformations*) è il successore di XSL, ed estende il concetto di foglio di stile fino a **permettere la manipolazione della struttura stessa del documento**.

XSLT permette **di trasformare un documento XML filtrandone i dati e riorganizzandoli in un'altra struttura XML**, HTML o persino in un semplice testo.


Xpath

XPath è un linguaggio tramite il quale è possibile esprimere delle espressioni per individuare parti di un documento XML, ovvero i nodi.

XPath mette a disposizione una serie di funzioni per la manipolazione di stringhe, numeri e booleani, da utilizzare per operare sui valori o sugli attributi dei nodi.

Le espressioni definite da XPath per accedere ai nodi dell'albero prendono il nome di Location Path (percorsi di localizzazione).

La struttura un location path è composta da tre componenti:

axis::node-test[predicate].


Le componenti axis, node-test, predicate

La componente **axis** esprime la relazione di parentela tra il nodo cercato ed il nodo corrente. (child, descendant, parent…)

node-test: specifica il tipo o il nome del nodo da cercare (enquiry, comment, test…)

predicate: contiene zero o più filtri per specificare delle condizioni più selettive da applicare alla ricerca (*[nometag='valore']*)


La componente axis

Le relazioni di parentela principali che possono essere contenute in axis sono:

•**ancestor**: indica tutti i nodi antenati del nodo corrente, ovvero tutti i nodi che lo precedono nell'albero associato al documento XML;

•**attribute**: indica tutti gli attributi del nodo corrente;

•**child**: indica i nodi figli del nodo corrente;

•**descendant**: indica tutti i discendenti del nodo corrente, ovvero tutti i nodi che hanno seguono il nodo corrente nell'albero XML;

•**parent**: indica il nodo genitore del nodo corrente, ovvero quello che lo precede nell'albero;

•**self**: indica il nodo corrente.


Sintassi

Possiamo utilizzare due diverse notazioni:

Sintassi abbreviata: compatta, che permette la realizzazione di costrutti intuitivi e facilmente realizzabili.

Es: /A/B/C *(tutti gli elementi di C figli di B figli di A, che rappresenta la radice del documento XML)*

Sintassi espansa: più complessa, che contiene maggiori opzioni, in grado di specificare con più particolarità gli elementi.

Es: /child::A/child::B/child::C

Esempio sintassi complessa

```
<?xml version="1.0"?>
<rubrica>
  <persona>
    <nome>Mario</nome>
    <cognome>Rossi</cognome>
    <indirizzo>
      <via>via bianchi 1</via>
      <cap>00000</cap>
      <citta>Roma</citta>
    </indirizzo>
    <telefono>
      <telefono_fisso gestore="Abc">123456</telefono_fisso>
      <telefono_cellulare gestore="Def">987656412</telefono_cellulare>
    </telefono>
  </persona>
</rubrica>
```

child::nome ->Questa espressione seleziona tutti i nodo chiamati 'nome' figli del nodo corrente.

Esempio sintassi complessa

child::nome	Questa espressione seleziona tutti i nodo chiamati 'nome' che sono figli del nodo corrente.
*child::**	Seleziona tutti i nodi figli del nodo corrente
attribute::gestore	Seleziona l'attributo di nome 'gestore' del nodo corrente.
descendant::cognome [cognome='Rossi']	Seleziona tutti i nodi chiamati 'cognome' tra i nodi discendenti del nodo corrente, il cui valore è Rossi.


Espressioni abbreviate

Le espressioni abbreviate, sono una versione semplificata e compatta dei Location Path ed offrono un meccanismo più veloce.

Queste espressioni sono costituite da una lista di nomi di elementi del documento XML, separati da uno slash(/), e tale lista descrive il percorso per accedere all'elemento desiderato.

Espressioni abbreviate

/rubrica/persona/nome	Recupera i nodi chiamati 'nome' indicando il percorso assoluto per raggiungere il nodo desiderato, attraverso una lista di nodi separata da slash
//nome	ricerca i nodi chiamati 'nome', in tutto il documento, indipendentemente dalla loro posizione e dal loro livello sull'albero associato al documento XML.
/rubrica//via	Ricerca tutti i nodi chiamati 'via' a qualsiasi livello dell'albero purchè contenuti all'interno del nodo chiamato 'rubrica'.
*/rubrica/persona/**	Ricerca qualsiasi elemento figlio del nodo chiamato 'persona'.
//nome/@valuta	Ricerca l'attributo 'valuta' dell'elemento 'nome'.
//persona[nome='Mario']	questa espressione ricerca tutti i nodi 'persona' che hanno il tag 'nome' il cui valore è Mario


Funzioni per gestire i nodi

XPath mette a disposizione anche delle funzioni per gestire i nodi, le stringhe, i numeri e i booleani.

•**count(node-set):** restituisce il numero di nodi contenuti nell'insieme di nodi passato come argomento della funzione;

•**name(nodo):** restituisce il nome di un nodo;


Funzioni per gestire i nodi

- **position():** determina la posizione di un elemento all'interno di un insieme di nodi;
- **last():** indica la posizione dell'ultimo nodo di un'insieme di nodi;
- **id(valore):** seleziona gli elementi in funzione del loro identificatore;
- **concat(s1,...,sn):** restituisce una stringa risultato della concatenazione delle stringhe specificate tra gli argomenti di una funzione;


Funzioni per gestire i nodi

- **string(valore):** converte il valore dell'argomento in una stringa;
- **string-length(stringa):** ritorna la lunghezza della stringa passata come parametro;
- **substring(stringa,inizio,lunghezza):** restituisce una sotto-stringa della stringa passata come argomento;

Funzioni per gestire i nodi

- **ceiling(numero):** arrotonda il numero per eccesso;
- **floor(numero):** arrotonda il numero per difetto;
- **number(valore):** converte il valore dell'argomento in un numero;
- **sum(node-set):** esprime la somma di un insieme di valori numerici contenuti in un insieme di nodi.

Esempio completo

```
<?xml version="1.0"?>
 <listautenti>
  <account user="fabio">
   <mail>fabio@aaaa.com</mail>
   <nome>Fabio V.</nome>
   <principale>
    <indirizzo>
     <via>Via Vai 1</via>
     <cap>98100</cap>
     <citta>Messina</citta>
    </indirizzo>
    <telefoni>
     <fisso>090123456</fisso>
     <cellulare
gestore="aaa">3001234567</
cellulare>
    </telefoni>
   </principale>
   <altrirecapiti>
    <ufficio>
     <indirizzo>
      <via>Via di Qua, 2</via>
      <cap>98100</cap>
      <citta>Messina</citta>
     </indirizzo>
     <telefoni>
      <fisso>09078901</fisso>
      <fax>3001234567</fax>
     </telefoni>
    </ufficio>
   </altrirecapiti>
  </account>
 </listautenti>
```

listautenti/account//telefoni/* restituisce la lista di tutti i nodi all'interno del nodo telefoni, in questo caso fisso, cellulare e fax.

/listautenti/account//indirizzo/.. restituisce tutti i nodi che contengono un nodo indirizzo al loro interno (ufficio e principale)

L'impiego dell'Axis // fa sì che vengano individuati anche nodi di livelli differenti purché all'interno di account.


Trasformazione XSLT

La trasformazione viene realizzata da un XSLT Processor che riceve come input il file XML da trasformare, il file XSL con la definizione dello stylesheet da applicare e produce come output il file trasformato

Un file XSL è formato da una serie di template che contengono le regole di trasformazione dei tag del documento XML.

Questi template vengono applicati ai tag corrispondenti dal XSLT Processor in maniera ricorsiva nel corso della trasformazione.

Come esempio vediamo di definire un file XSL per visualizzare un file XML di esempio in una pagina HTML

Trasformazione XSLT

Il file XML di input

```
<?xml version="1.0"?>
<rubrica>
  <persona>
    <nome>Mario</nome>
    <cognome>Rossi</cognome>
    <indirizzo>
      <via>via bianchi 1</via>
      <cap>00000</cap>
      <citta>Roma</citta>
    </indirizzo>
    <telefono>
      <telefono_fisso>123456</telefono_fisso>
      <telefono_cellulare>987656412</telefono_cellulare>
    </telefono>
  </persona>
</rubrica>
```


Trasformazione XSLT

Il file XSL con la definizione dello stile per la creazione del file HTML

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/
XSL/transform" version="1.0">
 <xsl:template match="/">
  <html>
   <head>
    <title>Rubrica in versione HTML</title>
   </head>
   <body>
    <h1>Rubrica</h1>
    <xsl:apply-templates/>
   </body>
  </html>
 </xsl:template>
 <xsl:template match="persona">
  <h2> <xsl:value-of select="cognome"/>
<xsl:value-of select="nome"/> </h2>
  <ul>
  <li>Via: <xsl:value-of select="./indirizzo/via"/></li>
  <li>CAP: <xsl:value-of select="./indirizzo/cap"/></li>
  <li>Citta': <xsl:value-of select="./indirizzo/citta"/></li>
   <li>Telefono (fisso): <xsl:value-of select="./
telefono/telefono_fisso"/></li>
   <li>Telefono (cellulare): <xsl:value-of select="./
telefono/telefono_cellulare"/></li>
  </ul>
 </xsl:template>
</xsl:stylesheet>
```

Trasformazione XSLT

Le regole di trasformazione sono contenute all'interno degli elementi ***template*** e tramite l'attributo **match** possiamo specificare, utilizzando la sintassi XPath, il tag a cui si riferiscono queste regole.

Nell'esempio:
<xsl:template match="/"> contiene le regole di trasformazione dell'elemento root del file di input (l'elemento <rubrica>);
<xsl:template match="persona"> definisce le regole per la trasformazione degli elementi <persona>.

Il processore XSLT effettua il parsing del documento XML da trasformare e, per ogni nodo incontrato, ricerca il *template* appropriato all'interno del file XSL.

Trasformazione XSLT

Il risultato sarà:

```
<html>
  <head>
    <title>Rubrica in versione HTML</title>
  </head>
  <body>
    <h1>Rubrica</h1>
    <h2>Rossi Mario</h2>
      <ul>
        <li>Via: via bianchi 1</li>
        <li>CAP: 00000</li>
        <li>Citta': Roma</li>
        <li>Telefono (fisso): 123456</li>
        <li>Telefono (cellulare): 987656412</li>
      </ul>
  </body>
</html>
```