Azzolini Riccardo 2020-03-26

XPath data model e serializzazione XML

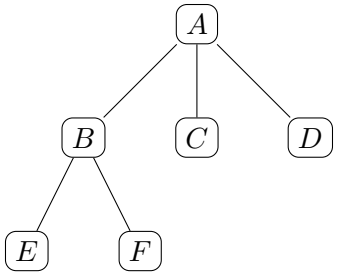
1 XPath Data Model

XPath Data Model è un modello dei dati semi-strutturato standardizzato dal W3C. Esso è stato definito come alternativa più semplice (e più adatta a essere utilizzata con il linguaggio d'interrogazione *XPath*) al più generale e complesso modello *XML Infoset*.

Nota: Come suggeriscono i loro nomi, questi modelli sono strettamente legati a XML, ma è importante non confonderli con XML "vero e proprio", che è solo uno dei modi per implementarli: in particolare, XML *non* è un modello dei dati.

1.1 Componenti del modello

A grandi linee, XPath Data Model è una "concettualizzazione" di un albero, in termini di elementi che lo compongono e relazioni tra di essi. Ad esempio, l'albero



è formato dai **nodi** A, B, C, D, E, F:

- il nodo A è la **radice**;
- i nodi B, C, D sono **figli** della radice A, ed E, F sono figli di B; viceversa, A è **padre** di B, C, D, e B è padre di E, F;
- i nodi C, D, E, F sono **foglie**, perché non hanno figli;
- tutti i nodi B, C, D, E, F sono **discendenti** di A (si dicono discendenti di un nodo i suoi figli, e i figli dei figli, ecc., fino alle foglie);
- i nodi B, C, D sono tra loro **fratelli** (in inglese *sibling*), così come lo sono E, F.

Inoltre, i nodi appartenenti a un albero possono essere di vari tipi:

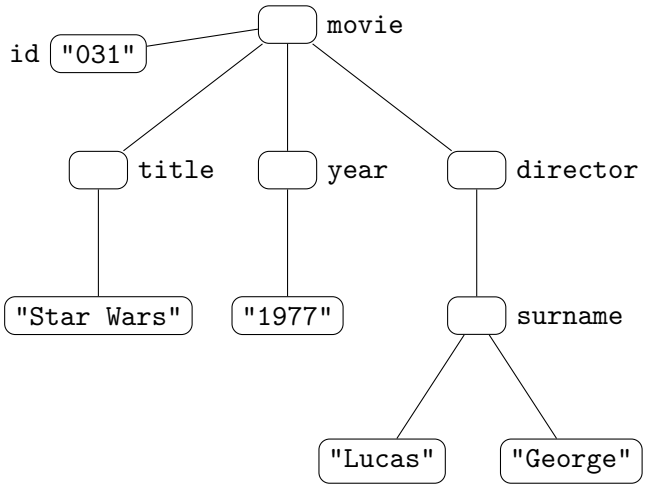
text: nodi foglia che contengono i dati "veri e propri";

element: nodi (solitamente) non terminali (cioè che hanno figli) ai quali è associato un nome;

attribute: rappresentano gli attributi del nodo element di cui sono figli;

processing: nodi foglia contenenti informazioni relative all'elaborazione con appositi strumenti (ad esempio per la formattazione / visualizzazione dei dati).

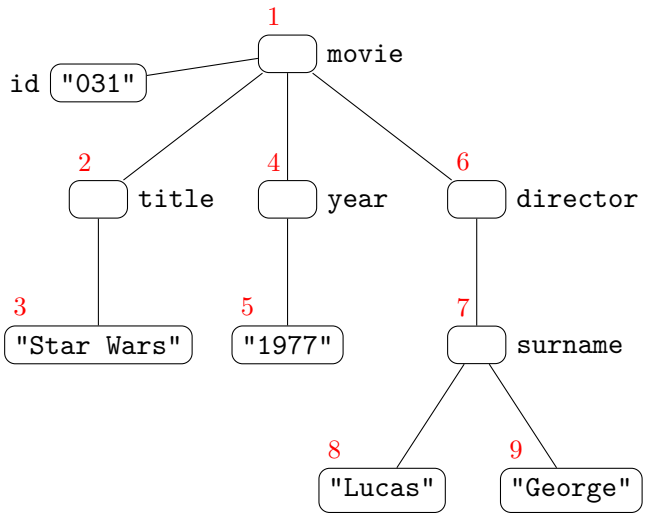
Ad esempio, nell'albero:



- i nodi `movie`, `title`, `year`, `directory`, `surname` e `name` sono di tipo *element*;
- `"Star Wars"`, `"1977"`, `"Lucas"` e `"George"` sono nodi di tipo *text*;
- `id` è un nodo di tipo *attribute*.

Questi alberi sono **ordinati**: l'ordine in cui i nodi vengono visitati è il **preordine da sinistra a destra**:[1]

[1] Per la precisione, XPath Data Model non include gli attributi nell'elenco dei figli di un elemento, e non ne considera l'ordine, perciò essi non sono direttamente inseriti nell'ordine di visita dell'albero.



2 Serializzazione XML

La **serializzazione** in XML di un albero XPath Data Model è una rappresentazione testuale (e quindi sequenziale, in quanto sequenza di caratteri) di tale albero.[2]

Note:

- In pratica, i termini "albero XPath", "albero XML" e "documento XML" vengono spesso usati come sinonimi.
- XML è soltanto una delle possibili serializzazioni di un modello semi-strutturato: in particolare, non è né la prima a essere stata definita, né la più semplice, ma è comunque uno degli standard de-facto per l'interscambio dei dati.

I componenti più elementari della serializzazione sono i caratteri Unicode, che vengono poi usati per creare:

- tag di markup (che rappresentano gli elementi);
- **character data** (i dati veri e propri, corrispondenti ai nodi di tipo text).

Ad esempio, un elemento con degli attributi viene serializzato come

```
<nome attributo-1="valore-1" ... attributo-n="valore-n">
  ...
</nome>
```

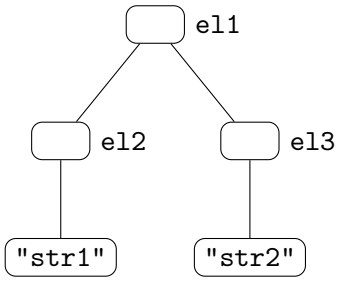
[2] La definizione per altri modelli dei dati semi-strutturati e per altri formati di serializzazione è analoga.

dove `<nome ...>` è detto **tag di apertura**, mentre `</nome>`, che indica la fine dell'elemento, è chiamato **tag di chiusura**.

Non tutti i documenti XML sono serializzazioni di alberi semi-strutturati. Infatti, perché una serializzazione sia **ben formata**, è necessario che i tag siano scritti nell'ordine di visita dell'albero. Ad esempio,

```
<el1><el2>str1</el2><el3>str2</el3></el1>
```

è una serializzazione ben formata, che corrisponde all'albero



mentre

```
<el1><el2>str1<el3>str2</el2></el3></el1>
```

non è una serializzazione ben formata, e perciò non rappresenta alcun albero.

Infine, alcune altre caratteristiche della serializzazione XML sono che:

- è *case sensitive* (a differenza, ad esempio, di HTML);
- è sempre presente, all'inizio del documento, una dichiarazione, chiamata *prologo*, che contiene informazioni di servizio necessarie per i parser[3] XML:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  ```

3 Namespace

All'interno di un documento XML, si potrebbero voler usare i tag di altri tipi di documento XML. Si potrebbe allora presentare il problema di tag diversi con lo stesso nome. Per risolverlo, XML mette a disposizione il meccanismo dei **namespace**.

Un namespace XML è identificato da un *URI* (*Uniform Resource Identifier*, una generalizzazione degli URI, Uniform Resource Locator). Ad esempio, i tag definiti per XHTML appartengono al namespace identificato da `http://www.w3.org/1999/xhtml`.

[3] Si chiama *XML parser* il componente software che traduce una serializzazione XML nel corrispondente albero XPath. Invece, il componente che effettua la traduzione nella direzione opposta è chiamato *XML serializer*.

Per poter usare i tag di un namespace, è necessario aggiungere a un elemento del documento XML un attributo speciale, come ad esempio:

```
<... xmlns:xh="http://www.w3.org/1999/xhtml">
```

esso indica l'uso del namespace XHTML, al quale si sceglie di associare l'abbreviazione `xh`. Così, all'interno dell'elemento dotato di tale attributo, i tag del namespace potranno essere usati aggiungendo ai loro nomi l'abbreviazione scelta (come prefisso, separato dal carattere "`:`"), in modo da evitare conflitti di nomi: ad esempio, bisognerà scrivere `<xs:head> ... </xs:head>` invece di `<head> ... </head>`, ma, se necessario, nello stesso documento potrà essere usato un altro tag `head` completamente indipendente da quello definito in XHTML.

Complessivamente, quindi, la sintassi per l'uso di un namespace è la seguente:

```
<... xmlns:xh="http://www.w3.org/1999/xhtml">
  <xh:head> ... </xh:head>
</...>
```

Osservazione: In pratica, la dichiarazione dell'uso di un namespace è analoga a un `import` in Java.