XML Schema

Fabio Vitali

Sommario

Oggi esaminiamo in breve XML Schema

- Perché non bastano i DTD
- Tipi ed elementi
- Definizione di elementi ed attributi
- Altri aspetti rilevanti di XML Schema

Motivazione (1)

Inizialmente si pensava che XML servisse solo per i documenti

XML è più semplice di SGML, è più generale ed aperto di HTML, è lo strumento ideale per esprimere documenti di testo, siano essi libri, manuali o pagine Web.

Quindi l'enfasi iniziale era su internazionalizzazione, strutturazione, facilità di conversione, ecc. Raccoglieva in pieno l'eredità di SGML. Lo sviluppo di XML era difatti condotto da membri della comunità SGML.

Motivazione (2)

Nasce poi l'idea che XML possa servire per qualcosa di più: XML è (anche) un linguaggio di markup per trasferire dati: un meccanismo per convertire dati dal formato interno dell'applicazione ad un formato di trasporto, facile da convertire in altri formati interni.

Non pensato per la visione umana, ma per essere prodotto ed usato da programmi.

XML è un'interfaccia (Adam Bosworth):

- Un'interfaccia tra autore e lettore, attraverso XSL e XLink, per portare significato tra creatore ed utente
- Un'interfaccia tra applicazione ed applicazione, attraverso XML Schema, per esprimere contratti sui formati, e verificarne il rispetto.

Motivazione (3)

Tutta la faccenda del trasferimento dei dati si semplifica: i documenti strutturati e gerarchici sono un formato ragionevole di sintassi praticamente per qualunque cosa: documenti di testo, record di database, ecc.

Nella W3C Note di Agosto 1999 (`http://www.w3.org/TR/schema-arch`)
"Many data-oriented applications are being defined which build their own data structures on top of an XML document layer, effectively using XML documents as a transfer mechanism for structured data; ”

Validazione e buona forma

La buona forma di un documento XML è una proprietà puramente sintattica.

La validazione, viceversa, è la verifica di un impegno preso **sopra** al formato, ad un livello già semantico:

- ***Mi impegno a scrivere dei documenti che siano formati da capitoli, ciascuno con un titolo e vari paragrafi, e ogni immagine con la sua didascalia.***

Per esprimere documenti di testo, i DTD probabilmente bastano, ma per esprimere blocchi di dati strutturati, è necessario un meccanismo di verifica più raffinato.

XML Schema è stato pensato per fornire quel supporto di validazione che i DTD permettono solo parzialmente, in particolare sul contenuto degli elementi e degli attributi del documenti XML.

XML Schema e DTD

I DTD non sono espressi con XML, così da dover creare strumenti appositi per la validazione.

I DTD non distinguono tra nome del tag e tipo del tag, ed hanno solo due tipi: complesso (cioè strutturato) e semplice (cioè CDATA o #PCDATA).

XML Schema, invece, fornisce un set complesso di tipi, a cui i tag e il loro contenuto debbono aderire.

Inoltre permette di agire in maniera object-oriented, permettendo di ampliare i tipi disponibili e di estenderne e precisarne le proprietà.

Infine, XML Schema è scritto in XML, permettendo l'uso di applicazioni XML per la verifica della validità dei dati espressi.

XML Schema

E’ una delle attività del working group su XML. Ha prodotto 7 generazioni di working draft, di cui l’ultima, del 24 ottobre 2000, dovrebbe essere pressoché definitiva.

E’ diviso in tre parti:

- XML Schema Part 0: Primer (un’introduzione)
- XML Schema Part 1: Structures (struttura del documento XML Schema)
- XML Schema Part 2: Datatypes (modello dei dati e meccanismi di estensione dei tipi)

Doveva essere una Recommendation W3C prima dell’estate. Poi sovrapposizioni di contenuto con XML-Query hanno ritardato lo sviluppo. Adesso è Proposed Recommendation.

Formato di un XML Schema

Un documento di XML Schema è racchiuso in un elemento <schema>, e può contenere, in varia forma ed ordine, i seguenti elementi:

- **`<import>`** ed **`<include>`** per inserire, in varia forma, altri frammenti di schema da altri documenti
- **`<simpleType>`** e **`<complexType>`** per la definizione di tipi denominati usabili in seguito
- **`<element>`** ed **`<attribute>`** per la definizione di elementi ed attributi **globali** del documento.
- **`<attributeGroup>`** e **`<group>`** per definire serie di attributi e gruppi di content model complessi e denominati.
- **`<notation>`** per definire notazioni non XML all'interno di un documento XML
- **`<annotation>`** per esprimere commenti per esseri umani o per applicazioni diverse dal parser di XML Schema.

I tipi in XML Schema

XML Schema usa i tipi per esprimere vincoli sul contenuto di elementi ed attributi.

- Un tipo semplice è un tipo di dati che non può contenere markup e non può avere attributi. In pratica è una sequenza di caratteri.
- Un tipo complesso è un tipo di dati che può contenere markup e avere attributi.

Esiste un grande numero di tipi predefiniti, di tipo semplice: string, decimal, float, boolean, uriReference, date, time, ecc.

Ogni tipo è caratterizzato da alcune proprietà, dette ***facets***, che ne descrivono vincoli e formati (permessi ed obblighi).

E' data possibilità di derivare nuovi tipi, sia per restrizione che per estensione di permessi ed obblighi.

Due esempi

```
<xsi:simpleType name='bodytemp’>
   <xsi:restriction base=’xsi:decimal'>
      <xsi:precision value=’3'/>
      <xsi:scale value='1'/>
      <xsi:minInclusive value=’36.5'/>
      <xsi:maxInclusive value=’44.0'/>
   </xsi:restriction>
</xsi:simpleType>

<xsi:complexType name='name'>
   <xsi:sequence>
      <xsi:element name='forename’ type=‘xsi:string’
          minOccurs='0’ maxOccurs=’unbounded'/>
      <xsi:element name='surname' type=‘xsi:string’/>
   </xsi:sequence>
</xsi:complexType>
```


Derivazione di tipi

I tipi possono formare un albero complesso di derivazioni, permettendo a nuovi tipi di attingere a definizioni di altri tipi.

E’ possibile estendere i tipi in due maniere:

- Per restrizione (<restriction>) ad esempio specificando facets aggiuntivi.
- Per estensione (<extension>) ad esempio estendendo il content model lecito. N.B.: l’estensione non ha senso per i tipi semplici.

Facets

Per ogni tipo io posso precisare dei facets, delle caratteristiche indipendenti tra di loro che specificano aspetti del tipo:

- length, minLength, maxLength: numero richiesto, minimo e massimo di caratteri
- minExclusive, minInclusive, maxExclusive, maxInclusive: valore massimo e minimo, inclusivo ed esclusivo
- precision, scale: numero di cifre significative e di decimali significativi
- pattern: espressione regolare che il valore deve soddisfare
- enumeration: lista all'interno dei quali scegliere il valore (simile alla lista di valori leciti degli attributi nei DTD).
- period, duration, encoding, ecc.

Esempi di derivazione (1)

```
<xsi:simpleType name='bodytemp’>
   <xsi:restriction base=’xsi:decimal'>
      <xsi:precision value=’3'/>
      <xsi:scale value='1'/>
      <xsi:minInclusive value=’36.5'/>
      <xsi:maxInclusive value=’44.0'/>
   </xsi:restriction>
</xsi:simpleType>

<xsi:simpleType name=’healthbodytemp’
   <xsi:restriction base=’bodytemp'>
      <xsi:maxInclusive value=’37.0'/>
   </xsi:restriction>
</xsi:simpleType>
```


Esempi di derivazione (2)

```
<xsi:complexType name='name'>
   <xsi:sequence>
      <xsi:element name='forename’ type=‘xsi:string’
           minOccurs='0’ maxOccurs=’unbounded'/>
      <xsi:element name='surname' type=‘xsi:string’/>
   </xsi:sequence>
</xsi:complexType>

<xsi:complexType name=’fullname’ >
   <xsi:extension base=“name”>
      <xsi:sequence>
         <xsi:element ref='title’ minOccurs='0'/>
      </xsi:sequence>
   </xsi:extension>
</xsi:complexType>
```


Definire elementi ed attributi

- Si usano gli elementi <element> e <attribute>.
- Se sono posti all'interno del tag schema sono elementi ed attributi ***globali*** (possono essere root di documenti).
- Altrimenti sono usabili solo all'interno di elementi che li prevedono come tipo.
- Questi hanno vari attributi importanti:
  - Name: il nome del tag o dell'attributo
  - Ref: il nome di un elemento o attributo definito altrove (globale)
  - Type: il nome del tipo, se non esplicitato come content
  - maxOccurs, minOccurs: il numero minimo e massimo di occorrenze
  - Fixed, default, nullable, ecc.: specificano valori fissi, di default e determinano la possibilità di elementi nulli.

Content model complessi (1)

Come nei DTD si usano virgole e caret per specificare obblighi e scelte tra gli elementi di un content model complesso, così in XML schema si usano <choice>, <sequence> e <all>. Questi sostituiscono anche le parentesi.

- (A, B) diventa

```
<xsi:sequence>
   <xsi:element ref=”A” />
   <xsi:element ref=”B” />
</xsi:sequence>
```

- (A | B) diventa

```
<xsi:choice>
   <xsi:element ref=”A” />
   <xsi:element ref=”B” />
</xsi:choice>
```


Content model complessi (2)

- (A, (B | C)) diventa

```
<xsi:sequence>
  <xsi:element ref=”A” />
  <xsi:choice>
    <xsi:element ref=”B” />
    <xsi:element ref=”C” />
  </xsi:choice>
</xsi:sequence>
```

- (A & B & C) (operatore di SGML) diventa

```
<xsi:all>
  <xsi:element ref=”A” />
  <xsi:element ref=”B” />
  <xsi:element ref=”C” />
</xsi:all>
```


Content model complessi (3)

Il content model misto aggiunge semplicemente l'attributo mixed con valore "true". Il content model vuoto viene specificato con un tipo complesso e privo di elementi. Un content model PCDATA con l'elemento "<simpleContent>". Così:

- (#PCDATA | B | C)* diventa

```
<xsi:complexType mixed="true">
   <xsi:element ref="B" />
   <xsi:element ref="C" />
</xsi:element>
```

- EMPTY diventa

```
<xsi:element name="vuoto">
    <xsi:complexType />
</xsi:element>
```


Attributi

La dichiarazione di attributi può avvenire con l'elemento <attribute>, dentro alla dichiarazione dell'elemento

```
<!ELEMENT A (B,C)>
<!ATTLIST A
          p   CDATA   #IMPLIED
          q   (uno|due|tre) #IMPLIED>
```

corrisponde a:

```
<xsi:element name="A">
  <xsi:complexType><xsi:sequence>
     <xsi:element ref="B" />
     <xsi:element ref="C" />
  </xsi:sequence></xsi:complexType>
  <xsi:attribute name="p" type="xsi:string"/>
  <xsi:attribute name="q"><xsi:simpleType>
    <xsi:restriction base="xsi:string">
       <xsi:enumeration value="uno"/>
       <xsi:enumeration value="due"/>
       <xsi:enumeration value="tre"/>
    </xsi:restriction></xsi:simpleType>
  </xsi:attribute>
</xsi:element>
```


Gruppi e gruppi di attributi

E’ possibile raccogliere i gruppi e gli attributi in gruppi:

```
<xsi:element name=”A”>
  <xsi:group ref=”elemA” />
 <xsi:attributeGroup ref=”attrsA” />
</xsi:element>
<xsi:group name=“elemA” />
 <xsi:complexType><xsi:sequence>
     <xsi:element ref=”B” />
     <xsi:element ref=”C” />
  </xsi:sequence></xsi:complexType>
</xsi:group>
 <xsi:attributeGroup name=”attrsA">
   <xsi:attribute name=”p” type=”xsi:string”/>
   <xsi:attribute name=”q"><xsi:simpleType>
    <xsi:restriction base="xsi:string">
       <xsi:enumeration value=”uno"/>
       <xsi:enumeration value=”due"/>
       <xsi:enumeration value=”tre"/>
    </xsi:restriction></xsi:simpleType>
  </xsi:attribute>
 </xsi:attributeGroup>
```


Annotazioni

Nei DTD l'unico posto dove mettere note e istruzioni di compilazione sono i commenti. Però i commenti sono a perdere: possono essere mangiati in qualunque fase dell'elaborazione. In XML Schema, invece, esiste un posto specifico dove mettere note ed istruzioni, l'elemento <annotation>.

L'elemento <annotation> può contenere elementi <documentation> (pensati per essere letti da esseri umani) oppure elementi <appInfo>, pensati per essere digeriti da applicazioni specifiche

```
<xsi:element name='pippo'>
  <annotation>
    <documentation>elemento pippo</documentation>
  </annotation>
  <xsi:element ref="pluto" />
</xsi:element>
```


I namespace (1)

La dichiarazione di targetNamespace definisce il namespace del documento da validare.

Gli attributi elementFormDefault e attributeFormDefault permettono di controllare se l'uso del prefisso è necessario per i tipi non globali.

```
<schema xmlns="http://www.w3.org/2000/08/XMLSchema"
        xmlns:po="http://www.example.com/PO1"
        targetNamespace="http://www.example.com/PO1"
        elementFormDefault="unqualified"
        attributeFormDefault="unqualified">
   <element name=”A”   type="po:prova"/>
   <element name=”C”   type="string"/>
   <complexType name=”prova">
     <sequence>
         <element name=“B” type=“string” >
         <element ref=“C” />
     </sequence>
   </complexType>
</schema>
```


I namespace (2)

Poiché gli attributi elementFormDefault e attributeFormDefault sono definiti come “unqualified”, solo i tipi globali debbono avere il prefisso, gli altri no.

```
<xx:A xmlns:xx="http://www.example.com/PO1">
   <B> ... </B>
   <xx:C> ... </xx:C>
</xx:A>
```

Alternativamente, si può usare il namespace di default e non usare mai prefissi:

```
<A xmlns="http://www.example.com/PO1">
   <B> ... </B>
   <C> ... </C>
</A>
```


I namespace (3)

Quello che i namespace permettono di fare è di specificare regole di validazione solo su alcuni e non tutti i namespace del documento:

```
<element name="htmlElement">
   <complexType>
      <sequence>
         <any namespace="http://www.w3.org/1999/xhtml"
              minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
</element>
```

Nell'attributo namespace dell'elemento <any> posso specificare o un namespace vero e proprio, o i valori:

- ##any: qualunque XML ben formato
- ##local: qualunque XML non sia qualificato (cioè privo di dichiarazione di namespace)
- ##other: qualunque XML tranne il target namespace

Altri aspetti

Inclusioni e importazioni

- In XML Schema, esistono meccanismi per dividere lo schema in più file, o per importare definizioni appartenenti ad altri namespace.
- **`<include schemaLocation="http://www. … "/>`**

Unicità e chiavi

- In XML Schema, è possibile richiedere che certi valori siano unici, o che certi valori siano chiavi di riferimenti, analoghi alla coppia ID/IDREF in XML “classico”. Tuttavia, è possibile specificare pattern molto complessi come elementi chiave.

Riferirsi ad uno schema

```
<fv:pippo xmlns:fv =“http://www.fabio.org/Pippo”
        xmlns:xsi=“http://www.w3.org/2000/08/XMLSchema”
        xsi:schemaLocation=“http://www.fabio.org/pippo.xsi”>
…
</fv:pippo>
```

Con l’attributo schemaLocation dentro all’istanza del documento XML diamo un suggerimento sulla posizione dello schema al validatore (ma la stessa informazione può essere data off-line, ad esempio perché predefinita, o in un header della connessione HTTP).

Conclusioni

Oggi abbiamo parlato di XML Schema:

- Motivazioni e status
- Organizzazione dei tipi
- Definizione di elementi ed attributi
- Content model, gruppi ed altri aspetti

Riferimenti

- D.C. Fallside ***XML Schema Part 0: Primer***, W3C Candidate Recommendation, 24 ottobre 2000, **http://www.w3.org/TR/xmlschema-0/**
- H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, ***XML Schema Part 1: Structures***, W3C Candidate Recommendation, 24 ottobre 2000, **`http://www.w3.org/TR/xmlschema-1/`**
- P. V. Biron, A. Malhotra, ***XML Schema Part 2: Datatypes***, W3C Candidate Recommendation, 24 ottobre 2000, **`http://www.w3.org/TR/xmlschema-2/`**
- Henry S. Thompson, ***XML Schema: An Intensive One-Day Tutorial***, WWW'99 Conference, **`ftp://ftp.cogsci.ed.ac.uk/pub/ht/tutorials/docs/Schema%20XML99.ppt`**
- R. L. Costello, ***XML Schemas***, **`http://www.xfront.com/xml-schema.zip`**