XML SCHEMA QUICK START

1. Introduzione

Il linguaggio XML Schema permette di definire le regole che deve soddisfare un particolare documento XML per poter essere considerato **valido**. La definizione di un XML Schema è specificata mediante un documento XML Schema Definition, in breve XSD. In generale un XSD viene utilizzato come segue:

```
<myTag xmlns="http://www.aaa.it"  xmlns:xsi=" http://www.bbb.it"
      xsi:schemaLocation="http://www.test.org/file.xsd">
```

L'attributo `xmlns` definisce il namespace di default, l'attributo `xmlns:xsi` indica il namespace associato al prefisso `xsi`. La dichiarazione del prefisso `xsi` è necessaria solo per poter usare l'attributo `schemaLocation` (che indica il file XSD dove viene effettivamente definito lo schema).

2. XML Schema: dati semplici

La definizione di un XML Schema ha sempre come root tag un elemento del genere:

```
<xs:schema xmlns:xs="http://www.aaa.it" targetNamespace=" http://www.bbb.it"
xmlns=" http://www.ccc.it" elementFormDefault="qualified">
```

L'attributo `xmlns:xs` indica il namespace dell'intero documento XSD (i.e. lo "schema dello schema"); l'attributo `targetNamespace` definisce il namespace degli elementi che verrano "regolamentati" dallo schema. In altre parole uno schema può definire un formato per delle tag dichiarate altrove: in questo modo diversi XML Schema possono essere definiti sullo stesso namespace (e poi usati in contesti diversi).
L'attributo `xmlns` indica il namespace di default (per le tag senza prefisso), mentre l'attributo `elementFormDefault` stabilisce se gli elementi definiti nello schema vanno qualificati (i.e. scritti con o senza prefisso).

Gli **elementi semplici** e gli **attributi** sono dichiarati come segue:

```
<xs:element name="xxx" type="yyy"/>
<xs:attribute name="xxx" type="yyy"/>
```

In ambo i casi il *type* può essere un tipo "base", ovvero: `xs:string`, `xs:decimal`, `xs:integer`, `xs:boolean`, `xs:date` oppure `xs:time`. E' possibile definire altri *types* tramite la tag

```
<xs:simpleType name="myType">…</xs:simpleType>
```

che può essere inserita direttamente all'interno di `element` o `attribute`, oppure altrove (in modo da poterla usare per più elementi o attributi). La tag `simpleType` permette di definire qualsiasi tipo di **restrizione** per il *data type*, utilizzando la sintassi `<xs:restriction base="xs:xxx">` e specificando delle *espressioni regolari* su diversi attributi: *enumeration*, *fractionDigits*, *length*, *pattern*, *totalDigits*, *whiteSpace* ecc.

Alcuni esempi di tag che possono essere indicate all'interno di `<xs:restriction>` :

`<xs:minInclusive value="0"/><xs:maxInclusive value="42"/>` : range di valori validi.

`<xs:enumeration value="item1"/><xs:enumeration value="item2"/>` : elenco di voci.

`<xs:pattern value="([A-Z][a-z])+"/>` : stringhe di almeno due caratteri con pattern "`Xx`".

3. XML Schema: dati complessi

Usando la tag `<xs:complexType>` è possibile definire dati complessi, al cui interno possono trovarsi altre tag, testo o tag alternate a testo. Scrivendo ad esempio

```
<xs:complexType name="customerData">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

è possibile usare `customerData` come `type` per le tag `element` e `attribute`.
Se il dato complesso contiene solamente testo e/o attributi va racchiuso da una tag `<xs:simpleContent>` (all'interno di `complexType`), che può contenere una tag `extension` oppure `restriction`:

```
<xs:extension base="xs:integer">
  <xs:attribute name="name" type="xs:string" />
</xs:extension>
```

Per aggiungere estensioni o restrizioni su un dato complesso è necessario inserire la tag `<xs:complexContent>` prima di definire la struttura dei dati (all'interno di `complexType`).
Per consentire l'uso di testo misto ad altre tag occorre specificare `<xs:complexType mixed="true">`, che permette di inserire dati del genere: `<outer>testo di esempio<inner>foo</inner></outer>`.

All'interno di `complexType` è possibile specificare l'**ordine** in cui devono apparire le tag interne, usando le tag `<xs:all>`, `<xs:choice>` oppure `<xs:sequence>`. Per limitare il **numero** di istanze di una tag figlio vanno usati gli attributi sull tag *element* in questione: `maxOccurs` e `minOccurs`. Siccome il valore di default è 1, per rendere opzionale una tag si scrive `minOccurs="0"`, per non limitare la sua occorenza si scrive `maxOccurs="unbound"`. Per rendere obbligatorio un attributo si scrive `use="required"`.
In alcuni casi risulta comodo definire un **gruppo** di tag o di attributi. La tag `<xs:group name="xxx">` permette di definire un gruppo di altre tag, solitamente racchiuse da una tag `<xs:sequence>`. La tag `<xs:attributeGroup name="yyy">` definisce invece un gruppo di attributi.
Una volta definiti, i gruppi vanno riferiti così:

```
<xs:sequence>
    <xs:group ref="myGroup"/>
    <xs:element name="myName" type="xs:string"/>
</xs:sequence>
```

Le tag `<any>` ed `<anyAttribute>` permettono invece di lasciare la possibilità di aggiungere tag e/o attributi non previsti dallo schema, in modo da rendere estensibile il formato. Infine è possibile **definire** degli alias per i nomi delle tag, ad esempio:

```
<xs:element                    name="name"                    type="xs:string"/>
<xs:element name="nome" substitutionGroup="name"/>
```

In questo modo la tag `<nome>` diventa a tutti gli effetti un sinonimo della tag `<name>` (supporto multilingua). E' possibile invalidare un alias (in un secondo tempo) inserendo `block="substitution"` all'interno di una tag del primo tipo (in questo caso la definizione di *name*). La sostituzione funziona solamente se le tag interessate sono **elementi globali**, ovvero sono introdotte direttamente come figli della root tag principale, che è sempre `<schema>`.

4. Data Types

Oltre al tipo `xs:string` già visto esistono altri tipi di **stringa**, ad esempio: `xs:normalizedString` indica che i caratteri di tabulazione, CR o LF inseriti dall'utente vengono convertiti nel numero equivalenti di *blank space*. Al contrario il tipo `xs:token` rimuove gli spazi prima e dopo il testo, e gli eventuali spazi multipli.
Altri tipologie di dati testuali sono: `ENTITIES`, `ENTITY`, `ID`, `IDREF`, `IDREFS`, `language`, `Name`, `NCName`, `NCName`, `NMTOKEN`, `NMTOKENS` e `QName`.

I **dati temporali** possono essere definiti usando i seguenti tipi:

- `xs:date` per le date nel formato YYYY-MM-DD.
- `xs:time` per gli orari nel formato HH:MM:SS.
- `xs:dateTime` per i timestamp nel formato YYYY-MM-DD**T**hh:mm:ss.
- `xs:duration` per gli intervalli di tempo: **P**YMDTHMS (usare `-P` per periodi negativi).

E' possibile specificare date e/o orari in UTC aggiungendo `Z` alla fine della data (`1970-26-02Z`), oppure indicando l'offset in ore rispetto all'orario UTC. Esempio: `<birthday>1970-26-02-01:00</birthday>` oppure `<birthday>1970-26-02+01:00</birthday>` (più o meno un'ora rispetto all'UTC).
Esistono anche altri tipi di dati che permettono di inserire solamente un giorno, un mese, un anno o altre combinazioni: `gDay`, `gMonth`, `gMonthDay`, `gYear` e `YearMonth`.

I tipi **numerici** possono essere: `byte`, `decimal`, `int`, `integer`, `long`, `negativeInteger`, `short`, `nonNegativeInteger`, `nonPositiveInteger`, `positiveInteger`, `unsignedLong`, `unsignedLong`, `unsignedInt`, `unsignedShort`, `unsignedByte`.

Altri tipi di dati sono gli `xs:base64Binary` e `xs:hexBinary` (binario ed esadecimale), i dati `double` e `float` per i numeri a virgola mobile, `anyURI` per indicare degli URI.
Infine, il tipo di dato `NOTATION` permette di specificare tipi di dati in formato non XML all'interno del documento XML, come ad esempio file o allegati associati a specifici programmi eseguibili.

E' possibile validare online un documento XMLSchema all'indirizzo:

http://www.w3.org/2001/03/webdata/xsv

5. Note

1. Facendo un'analogia con i database, è possibile pensare ad uno XMLSchema come alla *struttura* di una tabella (definizione delle colonne), mentre tutti i documenti XML che rispettano tale XMLSchema sono equivalenti ai *record* contenuti nella tabella.

2. Un foglio XMLSchema definisce la sintassi che un certo namespace deve rispettare, non la semantica. Se necessario, la semantica viene dichiarata a parte (*specifications*).

3. Se viene usato un `<xs:simpleType>` oppure un `<xs:complexType>` per definire un attributo, il *type* interno risulta non qualificato. Esempio:

   ```
   <xs:attribute name="foo" type="pippo">
   ```

   Il problema si risolve definendo come `xmlns` (namespace di default) lo stesso namespace specificato nel `targetNamespace`, in questo modo l'attributo `type="pippo"` viene identificato anche se sprovvisto di prefisso.

Appendice – Espressioni Regolari

Riportiamo di seguito alcuni esempi di espressioni regolari (*regular expression*) utili per definire i pattern

Espressione	Significato
`.`	Un carattere qualsiasi
`\w`	Un carattere alfanumerico
`\s`	Il carattere spazio (blank space)
`\d`	Un carattere numerico
`[A-Z]`	Un carattere dell'alfabeto maiuscolo
`[a-z]`	Un carattere dell'alfabeto minuscolo
`[A-F]`	I caratteri maiuscoli compresi tra A ed F
`[abc_]`	Solamente i caratteri *a*, *b*, *c* e *underscore*
`(…)*`	Zero o più occorrenze dell'espressione tra parentesi
`(…)?`	Zero o nessuna occorrenze dell'espressione tra parentesi (espressione opzionale)
`(…)+`	Minimo una (o più occorrenze) dell'espressione tra parentesi
`.{3}`	Qualsiasi stringa di esattamente 3 caratteri
`.{1,5}`	Qualsiasi stringa di di almeno 1 carattere e massimo 5 caratteri
`\d{6,13}`	Un numero di almeno 6 cifre e massimo 13 cifre
`(…)|(…)`	Indica un OR delle due espressioni (sono entrambe valide)

Note per i pattern XMLSchema:

1. Specificare più di un *pattern* all'interno della stessa tag **restriction** significa permettere l'OR logico tra tutti i pattern indicati (va bene uno qualsiasi di loro).

2. Specificando invece diverse *restrizioni* all'interno stessa tag **restriction** significa richiedere l'AND logico tra tutte le restrizioni indicate (devono essere soddisfatte tutte).

