La struttura: XML Schema

Laura Farinetti, Fulvio Corno
Dip. Automatica e Informatica
Politecnico di Torino

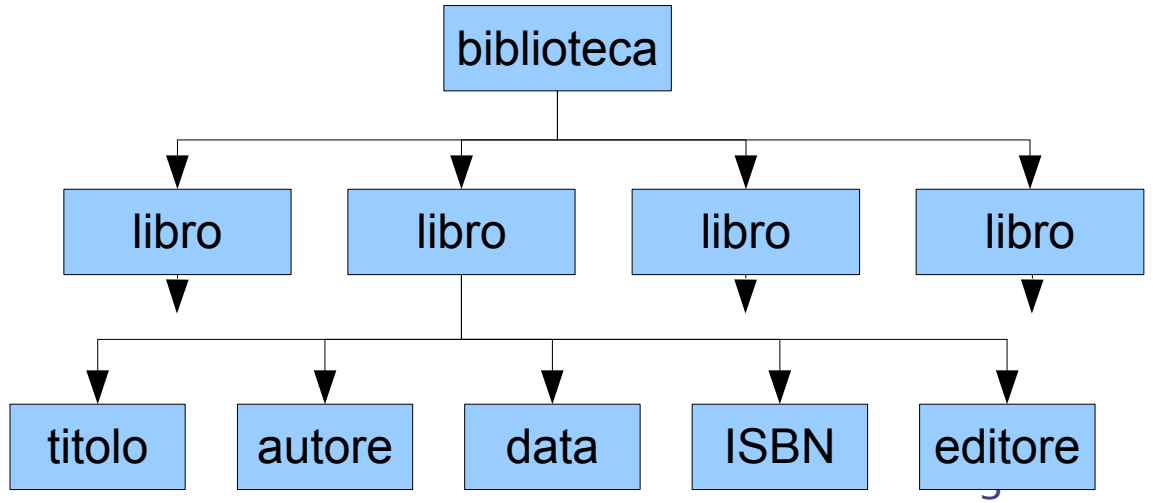
XML Schema

- Inizialmente proposto da Microsoft, e poi adottato dal W3C
- Alternativo ai DTD per specificare la struttura di un documento (grammatica)
- Recommendation del 2 maggio 2001, in tre parti
  - Primer (Part 0): descrizione generale
  - Structure (Part 1): linguaggio di descrizione degli schema
  - Datatypes (Part 2): definizione dei tipi di dato
- Sigla: XSD (Xml Schema Definition)

Esempio di XML Schema

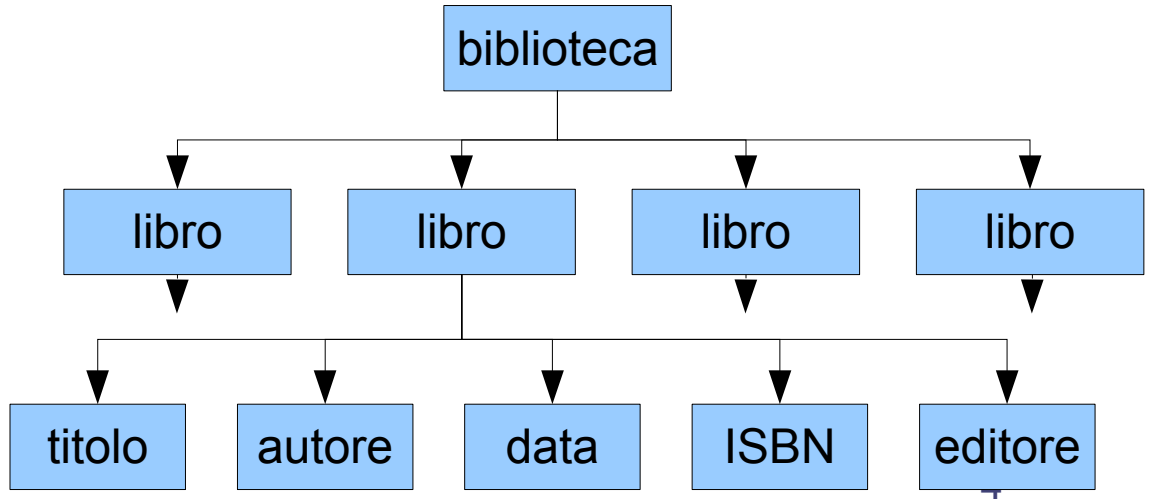
```
<?xml version="1.0"?>
<xsd:schema
      xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">

  <xsd:element name="biblioteca">
    <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="libro" minOccurs="1"
                                  maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```



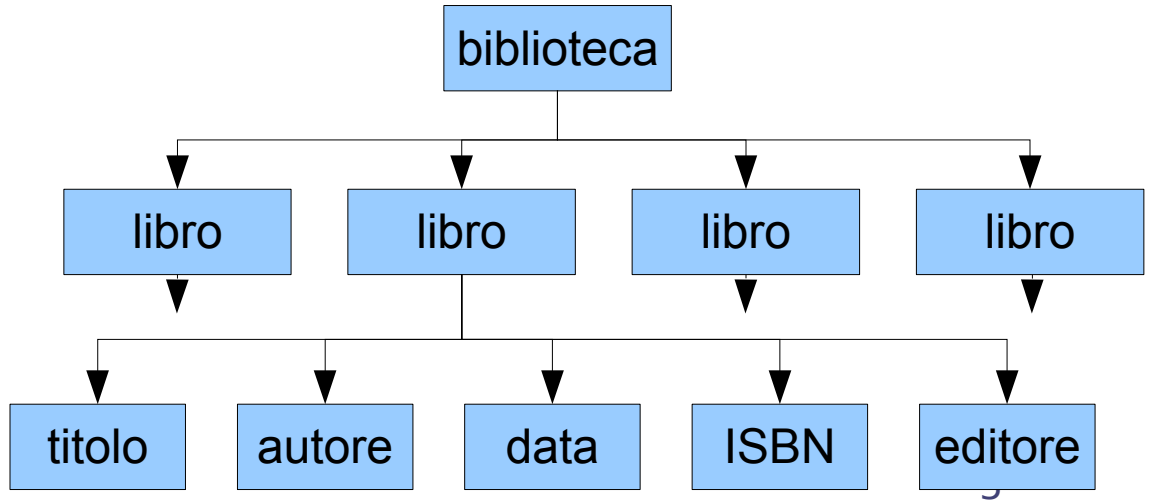
Esempio di XML Schema

```
<xsd:element name="libro">
  <xsd:complexType>
    <xsd:sequence>
       <xsd:element ref="titolo" minOccurs="1" maxOccurs="1"/>
       <xsd:element ref="autore" minOccurs="1"
            maxOccurs=" unbounded "/>
       <xsd:element ref="data" minOccurs="1" maxOccurs="1"/>
       <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
       <xsd:element ref="editore" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



Esempio di XML Schema

```
  <xsd:element name="titolo" type="xsd:string"/>
  <xsd:element name="autore" type="xsd:string"/>
  <xsd:element name="data" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="editore" type="xsd:string"/>
</xsd:schema>
```



XML Schema

- Maggiori potenzialità, e maggiore complessità
- Esempio: tipi di dato
  - built-in datatype
  - restrizioni
  - nuovi tipi di dato

```
<xsd:element name="titolo" type="xsd:string"/>
```

Associare uno Schema ad un documento

- È necessaria una *schema declaration* (analoga alla DTD declaration), all'interno del root element

```
<tns:GolfCountryClub
xmlns:tns="http://www.example.org/GolfCountryClub“
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance“
xsi:schemaLocation="http://www.example.org/GolfCountryClub
GolfCountryClub.xsd">
```

Tipi di elementi

- Simple Type
  - Elements that contain numbers (and strings, and dates, etc.) but do not contain any subelements
  - All attributes have simple types
- Complex Type
  - Elements that contain subelements or carry attributes

Built-in (simple) datatypes

```
<xsd:element name="data" type="xsd:string"/>
```

```
<xsd:element name="data" type="xsd:gYear"/>
```

- gYear: Gregorian calendar year
- Segue il formato: CCYY
  - range di CC: 00-99
  - range di YY: 00-99
  - esempio: 1999

Restrizioni

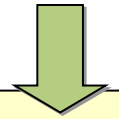
```
<xsd:element name="aircraft" default="Boeing 747">
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="Boeing 747"/>
         <xsd:enumeration value="F-16"/>
         <xsd:enumeration value="A-10"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

```
<!-- Usa il valore di default -->
<aircraft/>

<!-- Non usa il valore di default
<aircraft>F-16</aircraft>
```

Nuovi tipi di dato

```
<xsd:element name="ISBN" type="xsd:string"/>
```



```
<xsd:simpleType name="TipoISBN">
  <xsd:restriction base="xsd:string">
  <xsd:length value="13"/>
    <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>
    <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>
    <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>
  </xsd:restriction>
</xsd:simpleType>
...
...
<xsd:element name="ISBN" type="TipoISBN"/>
```

Nuovi tipi di dato

```
<xsd:element name="ISBN" type="xsd:string"/>
```

```
<xsd:simpleType name="TipoISBN">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>
    <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>
    <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>
  </xsd:restriction>
</xsd:simpleType>
...
...
<xsd:element name="ISBN" type="TipoISBN"/>
```

3 pattern in alternativa

uso delle “espressioni regolari”

Livelli di correttezza

- Nel caso di XML schema: due tipi di validità
  - content model validity: controlla che l'ordine ed il livello di annidamento dei tag sia corretto (Part 1 delle specifiche)
  - datatype validity: controlla che le specifiche unità di informazione siano del tipo corretto, e che i valori ricadano nell'intervallo di validità

Struttura dello schema

- XML Schema è un documento XML a tutti gli effetti
  - Occorre dichiararlo con riferimento al corretto Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/GolfCountryClub"
xmlns:tns="http://www.example.org/GolfCountryClub">
```

Schema definition

- Root element: `<xsd:schema>`
- Associate the standard W3C schema definition, by using he xmlns attribute (namespace definition)
  - prefix “xsd” used for all schema-related tags
    - `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`
  - no prefix (not recommended)
    - `xmlns="http://www.w3.org/2001/XMLSchema"`

Tipi di direttive XSD

- Creazione di nuovi tipi
  - *Simple*
  - *Complex*
- Definizione di elementi e attributi
  - Struttura dei file XML corrispondenti
  - Ciascun elemento o attributo ha uno dei tipi precedentemente definiti

Elementi ed attributi

- `<xsd:element name="name" type="xsd:string"/>`
  - name
  - type
- `<xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>`

xsd:complexType

- Definire un nuovo tipo complesso
  - `<xsd:complexType name="nomeTipo">`
    - Specifica del tipo
  - `</xsd:complexType>`

Definizione di complex type

- <xsd:sequence>
  - Contiene altre definizioni di <xsd:element>
- <xsd:attribute>

Esempio

```
<xsd:complexType name="USAddress" >
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="street" type="xsd:string"/>
    <xsd:element name="city" type="xsd:string"/>
    <xsd:element name="state" type="xsd:string"/>
    <xsd:element name="zip" type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:NMTOKEN"
fixed="US"/>
</xsd:complexType>
```

Esempio

```
<xsd:complexType name="USAddress" >
  <xsd:sequence>
    <xsd:element name="name   type="xsd:string"/>
    <xsd:element name="stree  type="xsd:string"/>
    <xsd:element name="city" t  "xsd:string"/>
    <xsd:element name="state" t  xsd:string"/>
    <xsd:element name="zip" t
  </xsd:sequence>
  <xsd:attribute name="count
fixed="US"/>
</xsd:complexType>
```

Da questo momento potremo definire nuovi elementi il cui tipo sia USAddress

Esempio di utilizzo

```
<xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
    <xsd:element name="shipTo" type="USAddress"/>
    <xsd:element name="billTo" type="USAddress"/>
    <xsd:element ref="comment" minOccurs="0"/>
    <xsd:element name="items"  type="Items"/>
  </xsd:sequence>
  <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
```

Ri-uso di elementi

- È possibile ri-usare la definizione di un elemento con l'attributo ref=
  - <xsd:element name="comment" type="xsd:string"/>
  - ...
  - <xsd:complexType><xsd:sequence> ...
    - <xsd:element ref="comment" minOccurs="0"/>

Molteplicità di elementi

- Gli elementi XSD possono essere caratterizzati da vincoli di molteplicità
  - minOccurs="1" (default: obbligatorio)
  - minOccurs="0" (opzionale)
  - minOccurs="7" (raro, ma possibile)
  - maxOccurs="1" (default: non ripetibile)
  - maxOccurs="unbounded" (ripetibile a piacere)
  - maxOccurs="12" (raro, ma possibile)

Molteplicità di attributi

- Gli attributi non possono mai essere ripetuti
- All'interno di xsd:attribute si usa una sintassi diversa:
  - use="required"
  - use="optional" (default)
  - use="prohibited"

Dichiarazioni globali

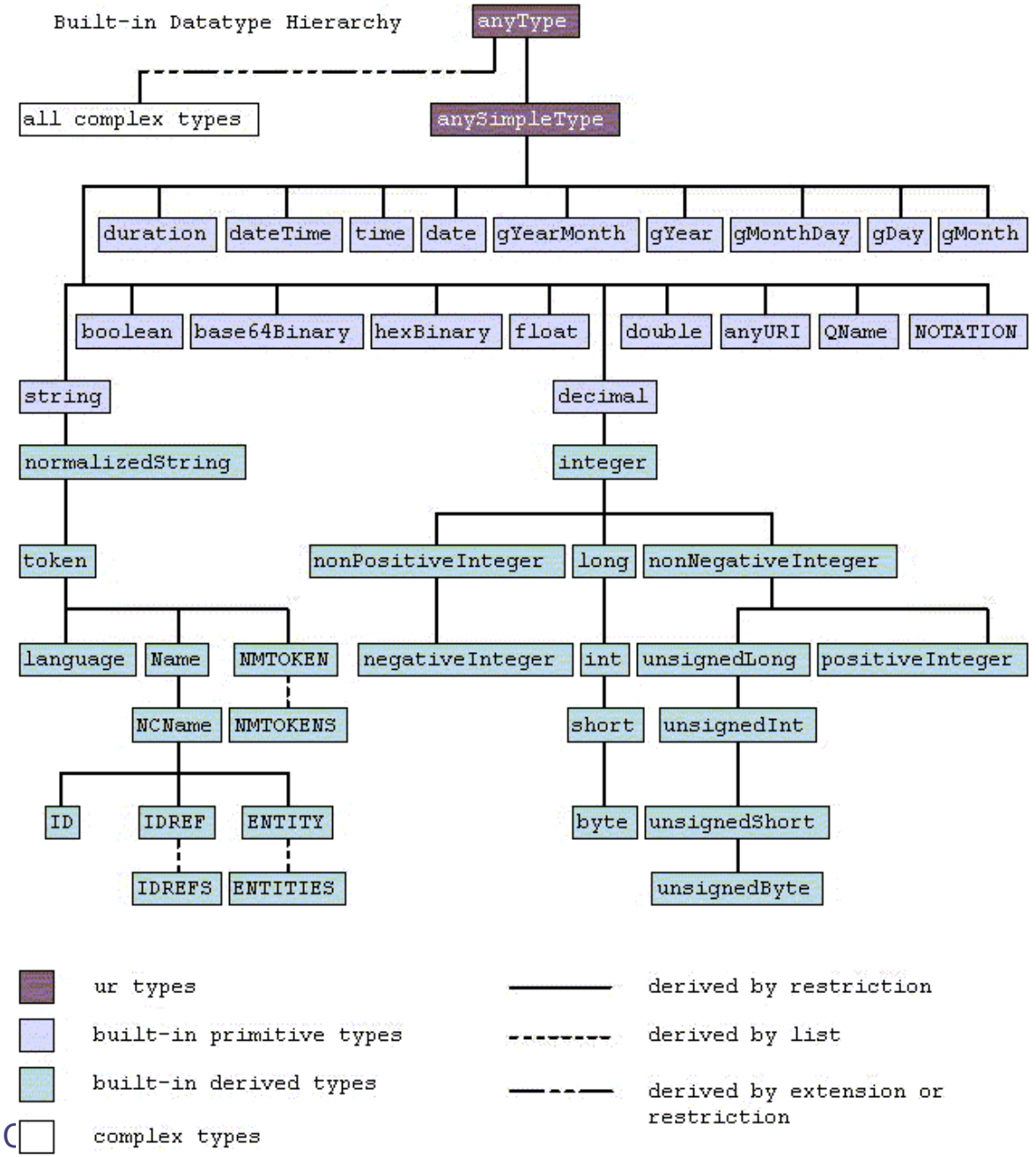
- Si definiscono “globali” gli elementi e gli attributi definiti direttamente come figli di <schema>
- Le dichiarazioni globali:
  - possono essere oggetto di un richiamo ref=””
  - possono essere il root element dei documenti XML collegati
  - non possono a loro volta contenere ref=
  - non possono contenere vincoli di cardinalità

Definizione di Simple Type

- Simple Type predefiniti
- Creazione di nuovi tipi mediante
  - Derivazione da altri simple type (predefiniti o no)
  - Applicazione di restrizioni, ossia selezione di sotto-insiemi di valori
- <xsd:restriction base="xsd:integer"> ...

Built-in Datatype

- Datatype primitivi
  - string, boolean, decimal, float, double, duration, dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth, hexBinary, base64Binary, anyURI, …
- Datatype derivati (sottotipi dei primitivi)
  - normalizedString, token, language, integer, nonPositiveInteger, negativeInteger, positiveInteger, nonNegativeInteger, long, int, short, byte, …


Built-in Datatype Hierarchy
anyType
all complex types
anySimpleType
duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
boolean
base64Binary
hexBinary
float
double
anyURI
QName
NOTATION
string
decimal
normalizedString
integer
token
nonPositiveInteger
long
nonNegativeInteger
language
Name
NMTOKEN
negativeInteger
int
unsignedLong
positiveInteger
NCName
NMTOKENS
short
unsignedInt
ID
IDREF
ENTITY
byte
unsignedShort
IDREFS
ENTITIES
unsignedByte
ur types
built-in primitive types
built-in derived types
complex types
derived by restriction
derived by list
derived by extension or restriction

Meccanismo di restrizione

- <xsd:restriction
  - base=”simpleTypeDiPartenza”>
  - criteri di restrizione <xsd:
    - length, minLength, maxLength, pattern, enumeration, whiteSpace, maxInclusive, maxExclusive, minInclusive, minExclusive, totalDigits, fractionDigits
  - >
- >

Tipi di restrizioni

- Intervalli di valori
  - <xsd:minInclusive value="10000"/>
    <xsd:maxInclusive value="99999"/>
- Pattern (regex)
  - <xsd:pattern value="\d{3}-[A-Z]{2}"/>
- Enumeration
  - <xsd:enumeration value="AK"/>
    <xsd:enumeration value="AL"/>
    <xsd:enumeration value="AR"/>

Esempi

```
<xsd:simpleType name="myInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="10000"/>
    <xsd:maxInclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
</xsd:simpleType>
```

Regular expression nei Pattern (1)

- a \| b alternativa
- (a) parentesi per impostare precedenze
- a* 0 o più volte
- a? opzionale
- a+ 1 o più volte
- a{n,m} da n ad m volte
- a{n} esattamente n volte
- a{n,} almeno n volte

Regular expression nei Pattern (2)

- [abc] lettera a, b oppure c
- [^abc] qualsiasi carattere tranne a, b, c
- [a-z] qualsiasi lettera minuscola
- I caratteri speciali vanno protetti con \ (esempio: \[)

Gruppi “speciali”

- . [^\n\r]
- \s spaziatura
- \S non spaziatura
- \d cifre [0-9]
- \D non cifre
- \w caratteri letterali (anche internazionali), eccetto spazi e punteggiatura

Esempio 1

- Consideriamo:
  - <internationalPrice currency="EUR"> 423.46</internationalPrice>
- Non può essere un Simple Type poiché ha un attributo
  - Definiamo un complex type

Esempio 1

```
<xsd:element name="internationalPrice">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="currency" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Esempio 1

```
<xsd:element name="internationalPrice">
  <xsd:complexType> <!-- definizione di tipo complesso anonimo -->
    <xsd:simpleContent> <!-- crea un tipo complesso da un tipo semplice-->
      <xsd:extension base="xsd:decimal"> <!-- contenitore di attributi
        <xsd:attribute name="currency" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Esempio 2

- Consideriamo:
  - <internationalPrice currency="EUR" value="423.46"/>
- L'elemento questa volta è vuoto, ma occorre comunque definire un complex type per specificare gli attributi

Esempio 2 (versione completa)

```
<xsd:element name="internationalPrice">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:restriction base="xsd:anyType"> <!-- restrizione fittizia -->
        <xsd:attribute name="currency" type="xsd:string"/>
        <xsd:attribute name="value"    type="xsd:decimal"/>
      </xsd:restriction>
    </xsd:complexContent> <!-- ...ma nessun xsd:element annidato!!! -->
  </xsd:complexType>
</xsd:element>
```

Esempio 2 (versione compatta)

```
<xsd:element name="internationalPrice">
  <xsd:complexType>
    <xsd:attribute name="currency" type="xsd:string"/>
    <xsd:attribute name="value"    type="xsd:decimal"/>
  </xsd:complexType>
</xsd:element>
```

Confronto con DTD

- Finora vi sono due caratteristiche dei DTD che non siamo ancora in grado di rappresentare:
  - l'alternativa (a|b) di elementi
  - la ripetizione a piacere in qualsiasi ordine (a|b|c)*
- A ciò servono le direttive xsd:group, xsd:choice, xsd:all

Esempio

```
<xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:group   ref="shipAndBill"/>
      <xsd:element name="singleUSAddress" type="USAddress"/>
    </xsd:choice>
    <xsd:element ref="comment" minOccurs="0"/>
    <xsd:element name="items"  type="Items"/>
  </xsd:sequence>
  <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>

<xsd:group id="shipAndBill">
  <xsd:sequence>
    <xsd:element name="shipTo" type="USAddress"/>
    <xsd:element name="billTo" type="USAddress"/>
  </xsd:sequence>
</xsd:group>
```

xsd:choice

- La direttiva xsd:choice permette la presenza di uno solo degli elementi dichiarati all'interno
- Uno, ed uno solo, deve essere presente
- Eventualmente, minOccurs=0 se si vuole rendere opzionale l'elemento

xsd:group

- Definisce un “gruppo” di elementi, che non costituiscono di per sé un elemento, ma vengono “riportati” dove serve per mezzo di ref=
- Fornisce un livello di controllo nell'xsd, senza introdurre elementi superflui nell'xml

xsd:all

- Definito come:
  - All the elements in the group may appear once or not at all, and they may appear in any order
- È una sorta di sequence non ordinata, ed in cui gli elementi sono opzionali
- Spesso associata a maxOccurs="unbounded"

Affermazione

- Named and un-named groups that appear in content models (represented by group and choice, sequence, all respectively) may carry minOccurs and maxOccurs attributes.
- By combining and nesting the various groups provided by XML Schema, and by setting the values of minOccurs and maxOccurs, it is possible to represent any content model expressible with an XML 1.0 DTD.
- Furthermore, the all group provides additional expressive power.

References

- XML Schema Primer (non normative)
  - http://www.w3.org/TR/xmlschema-0/
- XML Schema Part 1: Structures
  - http://www.w3.org/TR/xmlschema-1/
- XML Schema Part 2: Datatypes
  - http://www.w3.org/TR/xmlschema-2/