XML Schemas

http://www.w3.org/TR/xmlschema-0/ (Primer)
http://www.w3.org/TR/xmlschema-1/ (Structures)
http://www.w3.org/TR/xmlschema-2/ (Datatypes)

Roger L. Costello

Τεχνολογίες XML

Schema Validators

- Command Line Only
  - XSV by Henry Thompson
    - ftp://ftp.cogsci.ed.ac.uk/pub/XSV/XSV12.EXE
- Programmatic API
  - xerces by Apache
    - http://www.apache.org/xerces-j/index.html
  - IBM Schema Quality Checker (Note: this tool is only used to check your schema. It cannot be used to validate an instance document against a schema.)
    - http://www.alphaworks.ibm.com/tech/xmlsqc
  - MSXML4.0
    - http://www.microsoft.com
- GUI Oriented
  - XML Spy
    - http://www.xmlspy.com
  - Turbo XML
    - http://www.extensibility.com

Εισαγωγή 30”

- Στις επόμενες 3 διαφάνειες, θα γίνει μια πολύ γρήγορη και συνοπτική παρουσίαση των XML Schemas. Ο λόγος είναι για να γίνει κατανοητή η «συνολική εικόνα» πριν εξεταστούν οι λεπτομέρειες κατασκευής XML Schemas

Τι είναι τα XML Schemas?

- Απάντηση: Ένα λεξιλόγιο σε XML για την έκφραση των κανόνων που διέπουν τα XML δεδομένα

Παράδειγμα

```
<location>
        <latitude>32.904237</latitude>
        <longitude>73.620290</longitude>
        <uncertainty units="meters">2</uncertainty>
</location>
```

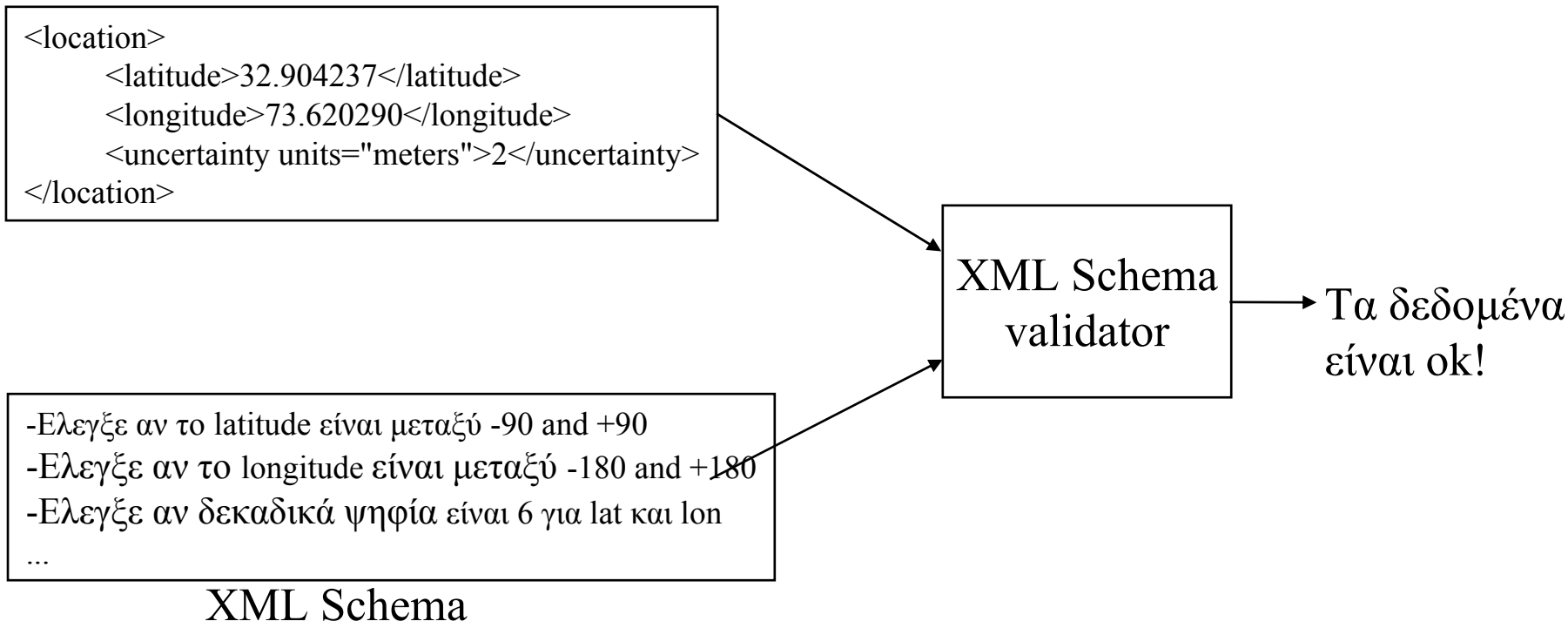
Είναι valid?

Για να είναι valid, πρέπει να ισχύουν οι εξής περιορισμοί **(data business rules)**:

1. Το location πρέπει να αποτελείται από latitude, ακολουθούμενο από longitude, ακολουθούμενο από το περιθώριο λάθους
2. Το latitude πρέπει να είναι δεκαδικός με τιμή μεταξύ -90 to +90
3. The longitude πρέπει να είναι δεκαδικός με τιμή μεταξύ -180 to +180
4. Τόσο για το latitude όσο και για το longitude ο αριθμός των δεκαδικών ψηφίων πρέπει να είναι ακριβώς 6
5. Η τιμή του περιθωρίου λάθους πρέπει να είναι ένας μη αρνητικός ακέραιος
6. Η μονάδα μέτρησης του περιθωρίου λάθους πρέπει να είναι μέτρα ή πόδια

Οι περιορισμοί μπορούν να εκφραστούν στα XML Schemas

Validating τα δεδομένα



Κυρίως πιάτο!

- Τώρα που είδατε την εισαγωγή 30’’, ας περάσουμε στις λεπτομέρειες!

Λόγος ύπαρξης των XML Schemas (και DTDs)

- Καθορισμός:
  - της *δομής* των XML εγγράφων
    - «αυτό το element περιέχει αυτά τα elements, που περιέχουν αυτά τα elements, κλπ"
  - τον *τύπο δεδομένων* κάθε element/attribute
    - «Αυτό το element πρέπει να είναι ακέραιος με τιμές από 0 εως 12,000" (Τα DTDs δεν τα καταφέρνουν τόσο καλά σε αυτό το ρόλο)

Κίνητρο για XML Schemas

- Δεν είμαστε ευχαριστημένοι με τα DTDs
  - Έχουν διαφορετική σύνταξη
  - Έχουν περιορισμένη δυνατότητα χειρισμού τύπων δεδομένων
    - Τα DTDs υποστηρίζουν ελάχιστους τύπους δεδομένων. Δεν μπορείς π.χ. να πεις «Θέλω το <elevation> να είναι ακέραιος με τιμές από 0 ως 12,000"
    - Επιθυμία για σύνολο τύπων δεδομένων συμβατό με τους αντίστοιχους που συναντάμε στις βάσεις
      - Τα DTD υποστηρίζουν 10 τύπους δεδομένων. Τα XML Schemas υποστηρίζουν 44+ τύπους δεδομένων

Highlights των XML Schemas

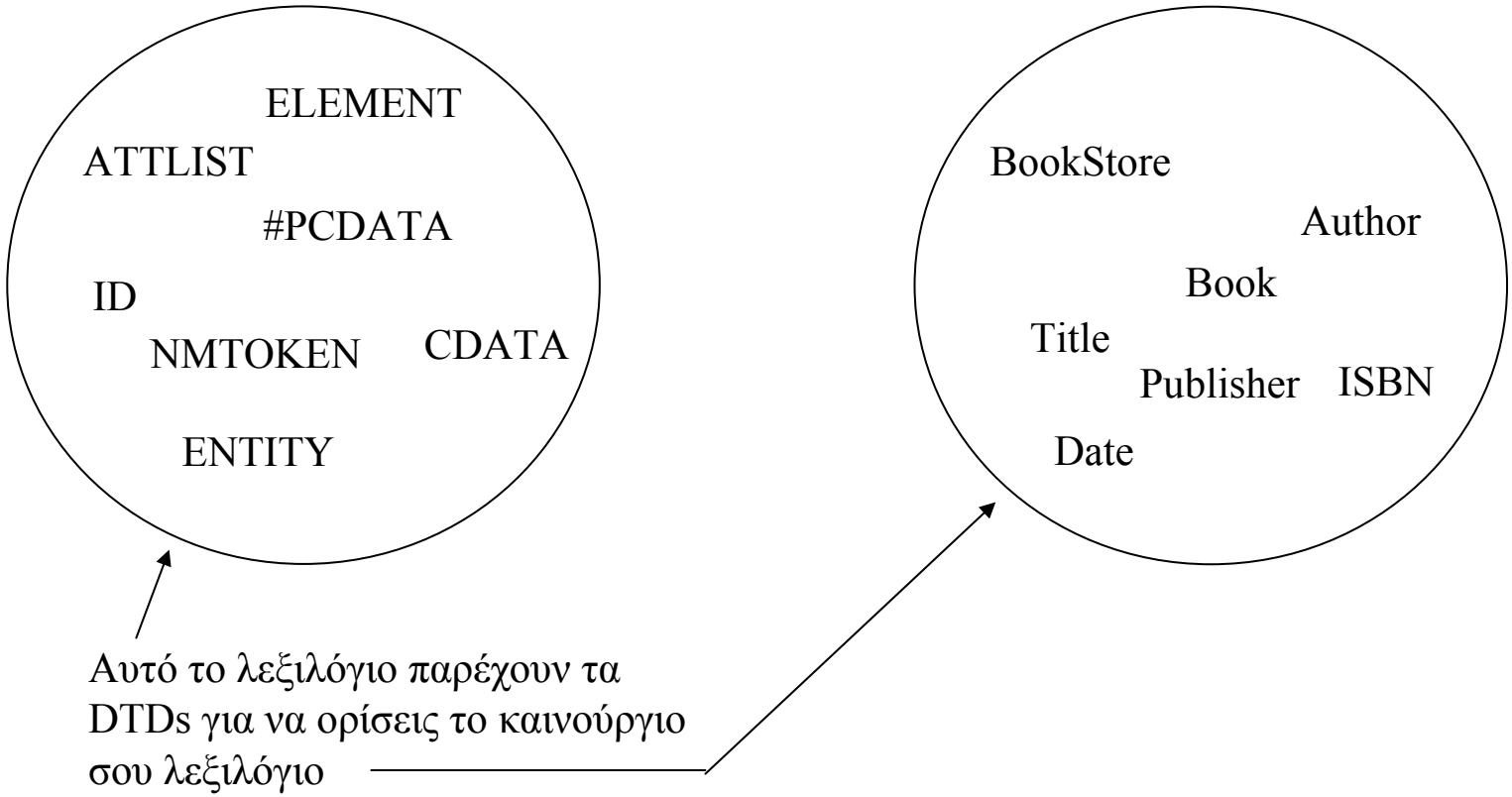
- Τα XML Schemas αποτελούν σημαντικότατη βελτίωση των DTDs:
  - Προηγμένοι τύποι δεδομένων
    - 44+ αντί 10
    - Δυνατότητα δημιουργίας προσωπικών τύπων δεδομένων
      - Παράδειγμα: «Αυτός είναι ένας νέος τύπος βασισμένος στον τύπο string ενώ τα συστατικά αυτού του τύπου πρέπει να ακολουθούν το μοτίβο : ddd-dddd, όπου 'd' αντιστοιχεί σε ψηφίο".
  - Εχουν την ίδια σύνταξη με τα XML
  - Αντικειμενοστραφή
    - Μπορούν να επεκτείνουν ή να περιορίσουν έναν τύπο
  - Μπορούν να ορίσουν σύνολα, δηλ., μπορούν να ορίσουν την ύπαρξη των child elements σε οποιαδήποτε σειρά
  - Μπορούν να ορίσουν το περιεχόμενο συστατικών να είναι μοναδικό όπως επίσης τη μοναδικότητα σε μια περιοχή
  - Μπορούν να ορίσουν πολλά elements με το ίδιο όνομα αλλά διαφορετικό περιεχόμενο
  - Μπορούν να ορίσουν elements χωρίς (nil) περιεχόμενο
  - Μπορούν να ορίσουν αντικαταστάσιμα elements – π.χ., το "Book" element μπορεί να αντικαταστήσει το"Publication" element.

Ας αρχίσουμε

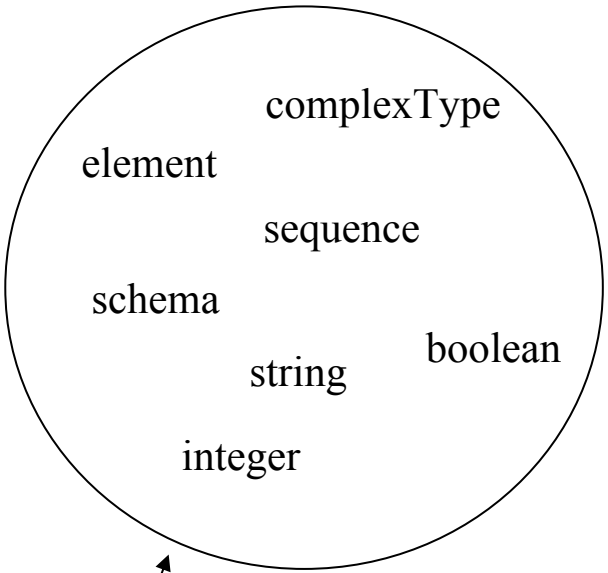
- Μετατροπή του BookStore.dtd (επόμενη σελίδα) σε σύνταξη XML Schema
  - Αρχικά θα κάνουμε μια απλή, 1-1 μετατροπή, δηλ., Title, Author, Date, ISBN, και Publisher θα περιέχουν strings, όπως και στο DTD
  - Σταδιακά, θα μεταβάλλουμε το XML Schema να χρησιμοποιεί πιο δυνατούς τύπους δεδομένων

BookStore.dtd

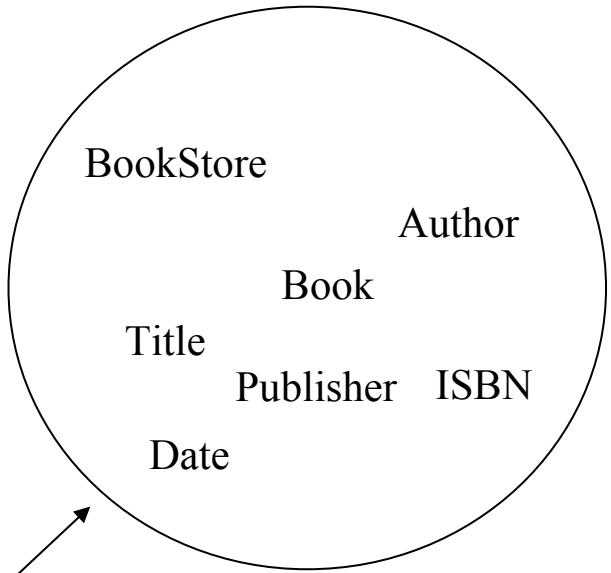
```
<!ELEMENT BookStore (Book+)>
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```


ELEMENT
ATTLIST
#PCDATA
ID
NMTOKEN
CDATA
ENTITY
BookStore
Author
Book
Title
Publisher
ISBN
Date
Αυτό το λεξιλόγιο παρέχουν τα DTDs για να ορίσεις το καινούργιο σου λεξιλόγιο

http://www.w3.org/2001/XMLSchema



http://www.books.org (*targetNamespace*)



Αυτό το λεξιλόγιο παρέχουν τα XML Schemas για να ορίσεις το καινούργιο σου λεξιλόγιο

Μια διαφορά μεταξύ XML Schemas και DTDs είναι ότι το λεξιλόγιο των XML Schema σχετίζεται με ένα όνομα (namespace). Αντίστοιχα, το νέο λεξιλόγιο που ορίζεις πρέπει να συσχετιστεί με ένα όνομα (namespace). Στα DTDs κανένα λεξιλόγιο δε σχετίζεται με κάποιο όνομα (namespace)

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

(εξηγήσεις στη συνέχεια)

BookStore.xsd

xsd = Xml-Schema Definition

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

<!ELEMENT BookStore (Book+)>

<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>

<!ELEMENT Title (#PCDATA)>

<!ELEMENT Author (#PCDATA)>

<!ELEMENT Date (#PCDATA)>

<!ELEMENT ISBN (#PCDATA)>

<!ELEMENT Publisher (#PCDATA)>

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Ολα τα XML Schemas έχουν το "schema" ως συστατικό ρίζα

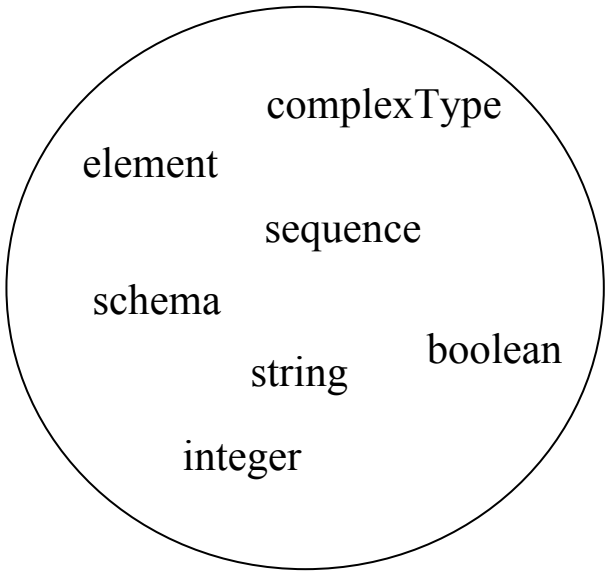
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Τα elements και οι τύποι δεδομένων που χρησιμοποιούνται για την κατασκευή
- schema
- element
- complexType
- sequence
- string

προέρχονται από το http://…/XMLSchema namespace

XMLSchema Namespace

http://www.w3.org/2001/XMLSchema



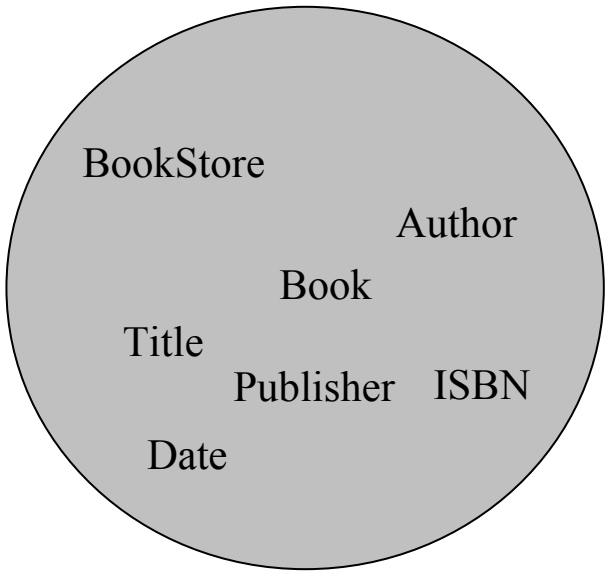
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Ορίζει ότι τα elements αυτού του schema
- BookStore
- Book
- Title
- Author
- Date
- ISBN
- Publisher

θα ανήκουν στο http://www.books.org namespace

Book Namespace (targetNamespace)

http://www.books.org (targetNamespace)



```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Το default namespace είνα το http://www.books.org που είναι το targetNamespace!

Αυτό αναφέρει μια Book element declaration To Book σε ποιό namespace? Αφού δεν υπάρχει namespace qualifier, αναφέρει το Book element στο default namespace, δηλ. το targetNamespace! Ετσι, αυτή είναι μια αναφορά στην Book element declaration σε αυτό το schema.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Αυτό είναι μια εντολή σε όλα τα instance documents που υπακούν σε αυτό το schema:
Όσα elements χρησιμοποιούνται από το instance document και ορίζονται σε αυτό το schema πρέπει να είναι namespace qualified.

Αναφέροντας ένα schema σε ένα XML έγγραφο

```
<?xml version="1.0"?>
<BookStore xmlns ="http://www.books.org" (1)
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" (3)
           xsi:schemaLocation="http://www.books.org
                               BookStore.xsd"> (2)
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>July, 1998</Date>
        <ISBN>94303-12021-43892</ISBN>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
    ...
</BookStore>
```

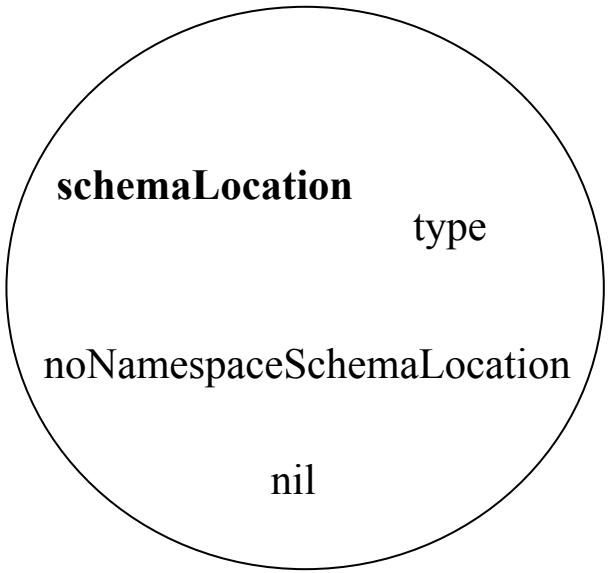
1. Πρώτον, η χρήση μιας default namespace δήλωσης, λέει στο schema-validator ότι όλα τα elements σ' αυτό το έγγραφο προέρχονται από το *http://www.books.org* namespace.

2. Δεύτερον, η δήλωση schemaLocation λέει στο schema-validator ότι το *http://www.books.org* namespace ορίζεται στο BookStore.xsd (δηλ., **schemaLocation περιέχει ζεύγος τιμών**).

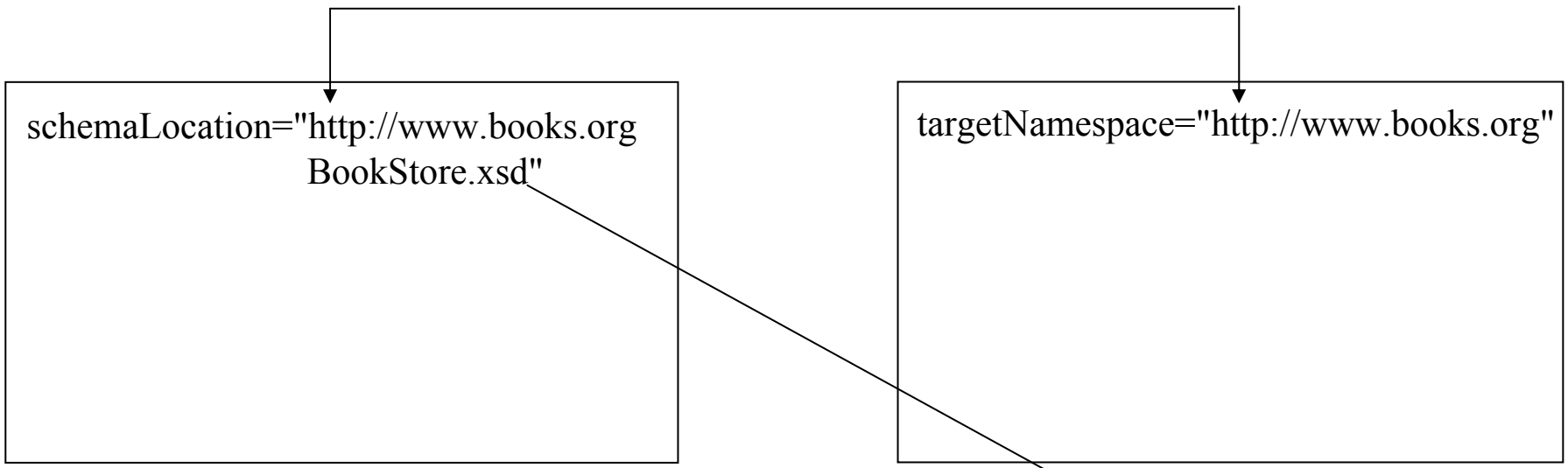
3. Τρίτον, πες στο schema-validator ότι το schemaLocation attribute που χρησιμοποιείται προέρχεται από το XMLSchema-instance namespace.

XMLSchema-instance Namespace

http://www.w3.org/2001/XMLSchema-instance

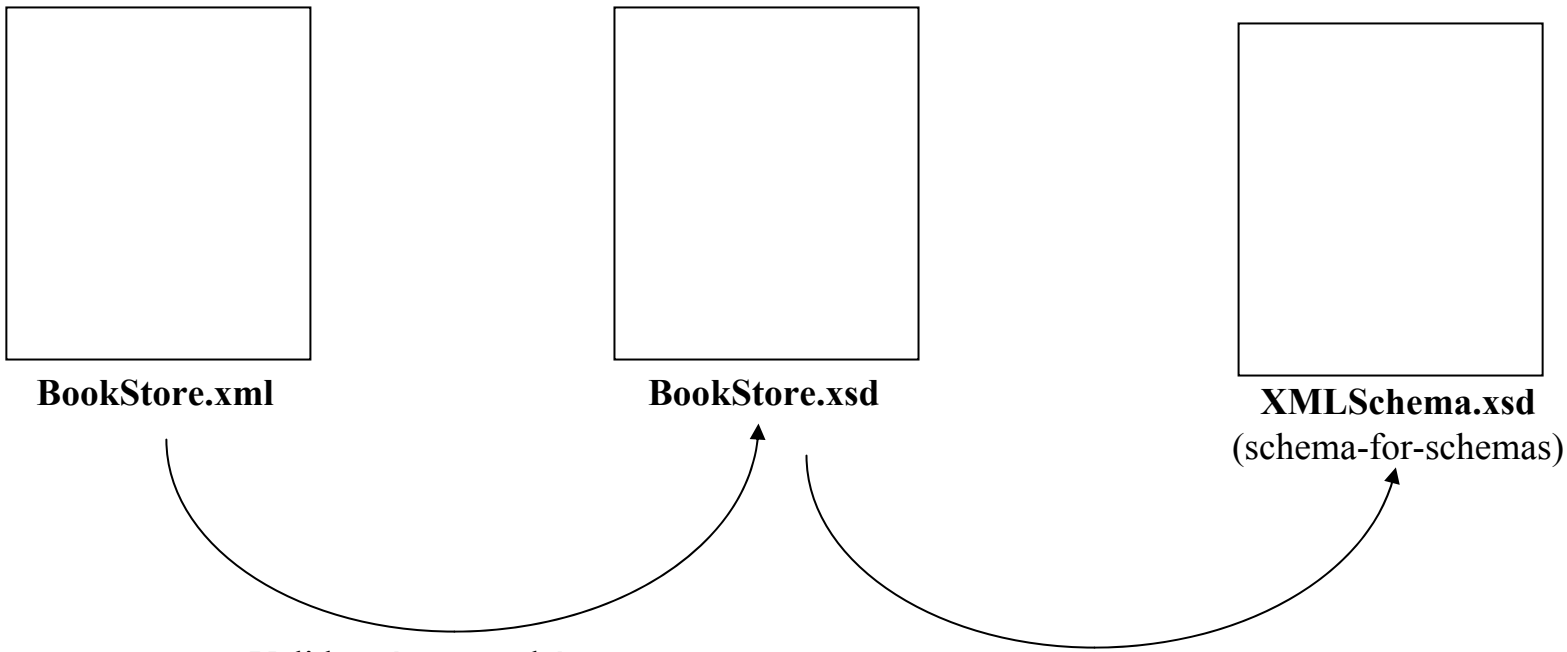


Αναφέροντας ένα schema σε ένα XML έγγραφο



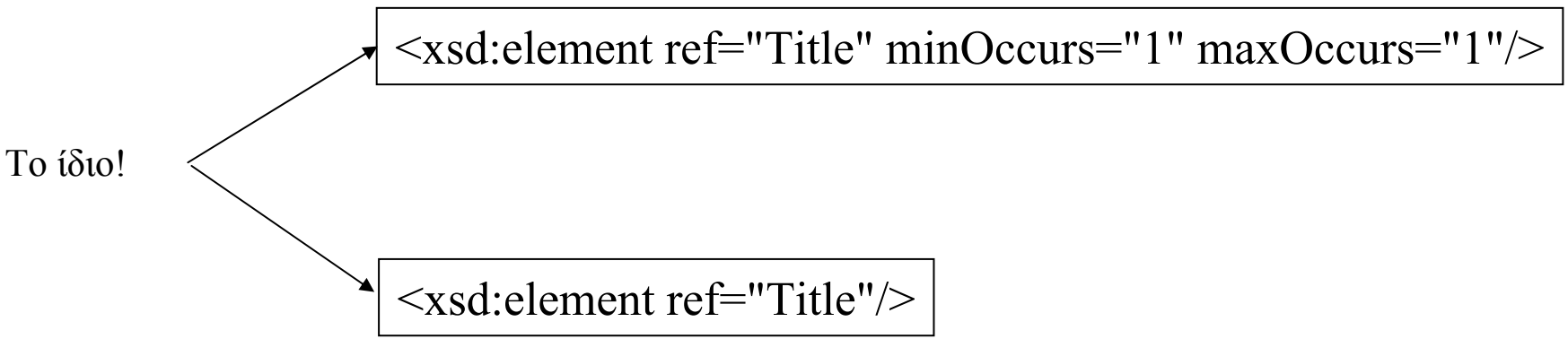
Το schema *ορίζει* ένα νέο λεξιλόγιο. Instance documents *χρησιμοποιούν το* νέο λεξιλόγιο

Πολλαπλά επίπεδα ελέγχου



Default τιμή για minOccurs και maxOccurs

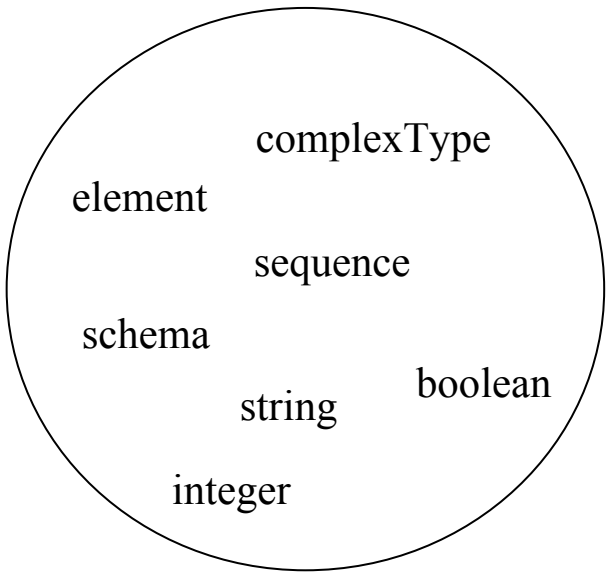
- Η default τιμή για minOccurs είναι "1"
- Η default τιμή για maxOccurs είναι "1"



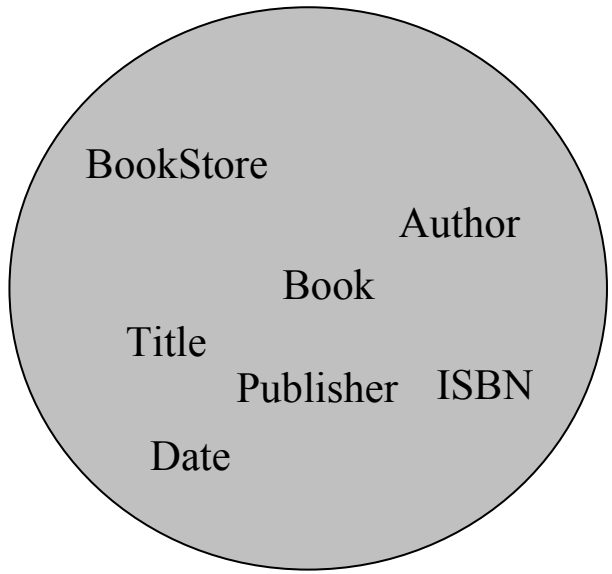
Qualify XMLSchema, Default targetNamespace

- Στο πρώτο παράδειγμα, εξειδικεύσαμε ρητά (explicitly qualified) όλα τα elements από το XML Schema namespace. Το targetNamespace ήταν το default namespace.

http://www.w3.org/2001/XMLSchema



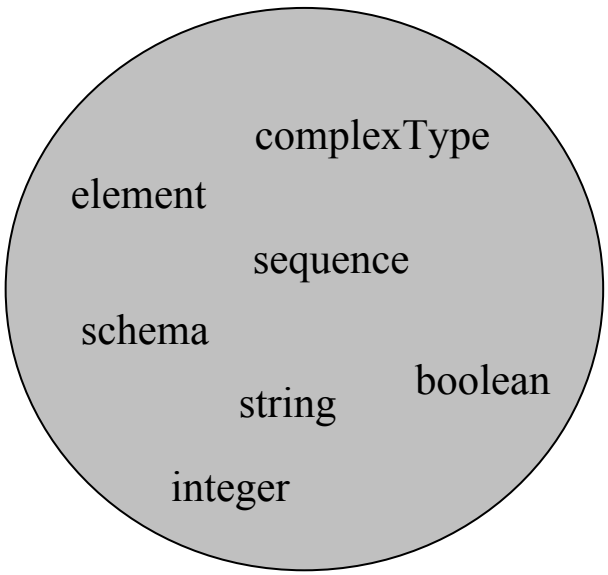
http://www.books.org (targetNamespace)



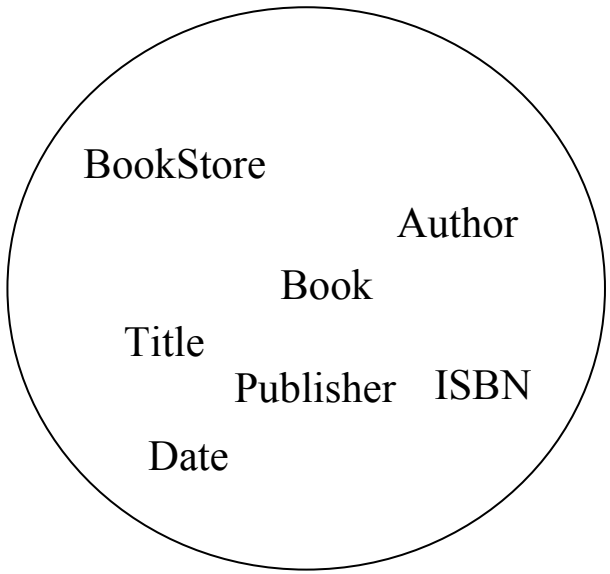
Default XMLSchema, Qualify targetNamespace

- Εναλλακτικά (όμοια), μπορούμε να σχεδιάσουμε το schema έτσι ώστε το XMLSchema να είναι το default namespace.

http://www.w3.org/2001/XMLSchema



http://www.books.org (targetNamespace)



```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns:bk="http://www.books.org"
            elementFormDefault="qualified">
   <element name="BookStore">
      <complexType>
         <sequence>
            <element ref="bk:Book" maxOccurs="unbounded"/>
         </sequence>
      </complexType>
   </element>
   <element name="Book">
      <complexType>
         <sequence>
            <element ref="bk:Title"/>
            <element ref="bk:Author"/>
            <element ref="bk:Date"/>
            <element ref="bk:ISBN"/>
            <element ref="bk:Publisher"/>
         </sequence>
      </complexType>
   </element>
   <element name="Title" type="string"/>
   <element name="Author" type="string"/>
   <element name="Date" type="string"/>
   <element name="ISBN" type="string"/>
   <element name="Publisher" type="string"/>
</schema>
```

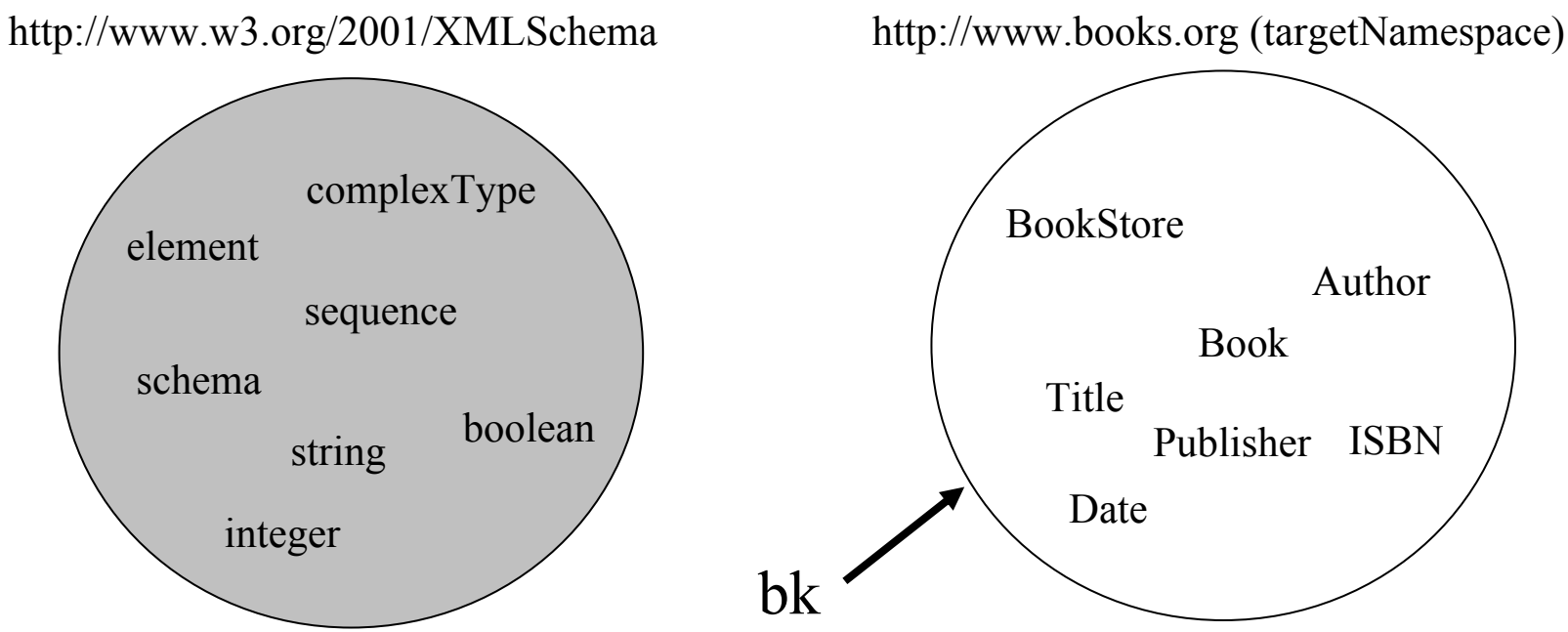
Το http://…/XMLSchema είναι το default namespace.
Συνεπώς, δεν υπάρχουν namespace qualifiers στα
- schema
- element
- complexType
- sequence
- string

(see example02)

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.books.org"
        xmlns:bk="http://www.books.org"
        elementFormDefault="qualified">
  <element name="BookStore">
    <complexType>
      <sequence>
        <element ref="bk:Book" minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>
  <element name="Book">
    <complexType>
      <sequence>
        <element ref="bk:Title"/>
        <element ref="bk:Author"/>
        <element ref="bk:Date"/>
        <element ref="bk:ISBN"/>
        <element ref="bk:Publisher"/>
      </sequence>
    </complexType>
  </element>
  <element name="Title" type="string"/>
  <element name="Author" type="string"/>
  <element name="Date" type="string"/>
  <element name="ISBN" type="string"/>
  <element name="Publisher" type="string"/>
</schema>
```

Εδω αναφέρουμε το Book element. Πού είναι ορισμένο το Book element? Σε ποιό namespace? Το πρόθεμα bk: υποδηλώνει σε ποιο namespace ανήκει αυτό το element. Το bk: ρυθμίστηκε να είναι το ίδιο με το targetNamespace.

"bk:" Αναφέρει το targetNamespace



συνεπώς, το, *bk:Book* αναφέρεται στο Book element στο targetNamespace.

Inlining δηλώσεις Elements

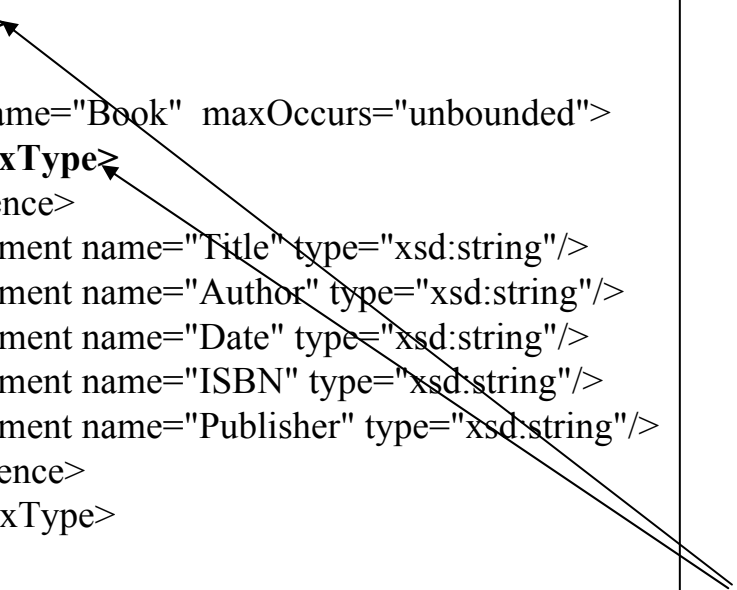
- Στα προηγούμενα παραδείγματα δηλώναμε ένα element και μετά αναφερόμαστε σ' αυτή τη δήλωση. Εναλλακτικά, μπορούμε να έχουμε *inline* δηλώσεις elements
- Στην επόμενη διαφάνεια υπάρχει ένας εναλλακτικός (όμοιος) τρόπος αναπαράστασης του schema που είδαμε πριν, χρησιμοποιώντας *inlined element δηλώσεις*

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book"  maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Title" type="xsd:string"/>
              <xsd:element name="Author" type="xsd:string"/>
              <xsd:element name="Date" type="xsd:string"/>
              <xsd:element name="ISBN" type="xsd:string"/>
              <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Εχουμε μεταφέρει
όλες τις δηλώσεις elements
inline, οπότε και δεν
αναφερόμαστε σε αυτές.
αυτό έχει ως αποτέλεσμα
τη δημιουργία πιο
συμπαγών schema!

Αυτός ο τρόπος σχεδιασμού schema - by inlining everything – ονομάζεται *σχεδιασμός Russian Doll*.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
   <xsd:element name="BookStore">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Book"  maxOccurs="unbounded">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="Title" type="xsd:string"/>
                     <xsd:element name="Author" type="xsd:string"/>
                     <xsd:element name="Date" type="xsd:string"/>
                     <xsd:element name="ISBN" type="xsd:string"/>
                     <xsd:element name="Publisher" type="xsd:string"/>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

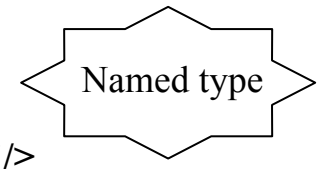


(example03)

Επώνυμοι τύποι

- Η επόμενη διαφάνεια δείχνει ένα εναλλακτικό (όμοιο) schema που χρησιμοποιεί ένα επώνυμο complexType

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" type="BookPublication" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="BookPublication">
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string"/>
      <xsd:element name="Date" type="xsd:string"/>
      <xsd:element name="ISBN" type="xsd:string"/>
      <xsd:element name="Publisher" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```



(example04)

Το πλεονέκτημα απομάκρυνσης των δηλώσεων των elements και η ενσωμάτωσή τους σε ένα επώνυμο τύπο είναι ότι αυτός ο τύπος μπορεί να επαναχρησιμοποιηθεί

Το παρακάτω σενάριο:

```
<xsd:element name="A" type="foo"/>
<xsd:complexType name="foo">
    <xsd:sequence>
        <xsd:element name="B" …/>
        <xsd:element name="C" …/>
    </xsd:sequence>
</xsd:complexType>
```

Το Element A *αναφέρει* το complexType foo.

είναι όμοιο με:

```
<xsd:element name="A">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="B" …/>
            <xsd:element name="C" …/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Το Element A έχει τον ορισμό του complexType *inlined* στη δήλωση του element

type Attribute ή complexType Child Element, αλλά όχι και τα 2!

- Μια δήλωση ενός element μπορεί να έχει το attribute type, ή ένα complexType child element, αλλά δεν μπορεί να έχει και τα δυο.

```
<xsd:element name="A" type="foo">
    <xsd:complexType>
        …
    </xsd:complexType>
</xsd:element>
```

Ανακεφαλαίωση δήλωσης Elements (2 τρόποι)

1

<xsd:element name="*name*" type="*type*" minOccurs="*int*" maxOccurs="*int*"/>

Ενας απλός τύπος (πχ., xsd:string) ή το όνομα ενός complexType (π.χ. BookPublication)

Ενας μη αρνητικός ακέραιος

Ενας μη αρνητικός ακέραιος ή "unbounded"

Σημ: minOccurs και maxOccurs μπορούν να χρησιμοποιηθούν μόνο σε τοπικές (local) δηλώσεις elements

---

2

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
    <xsd:complexType>
        …
    </xsd:complexType>
</xsd:element>
```

Built-in τύποι δεδομένων

- Primitive Datatypes
  - string
  - boolean
  - decimal
  - float
  - double
  - duration
  - dateTime
  - time
  - date
  - gYearMonth
  - gYear
  - gMonthDay

- Atomic, built-in
  - **"Hello World"**
  - {**true**, **false, 1, 0**}
  - **7.08**
  - **12.56E3**, **12**, **12560**, **0**, **-0**, **INF**, **-INF**, **NAN**
  - **12.56E3**, **12**, **12560**, **0**, **-0**, **INF**, **-INF**, **NAN**
  - **P1Y2M3DT10H30M12.3S**
  - format: *CCYY-MM-DDThh:mm:ss*
  - format: *hh:mm:ss.sss*
  - format: *CCYY-MM-DD*
  - format: *CCYY-MM*
  - format: *CCYY*
  - format: *--MM-DD*

Σημ: 'T' είναι ο διαχωριστής date/time
INF = infinity
NAN = not-a-number

Built-in τύποι δεδομένων (συν.)

- Primitive Datatypes
  - gDay
  - gMonth
  - hexBinary
  - base64Binary
  - anyURI
  - QName
  - NOTATION

- Atomic, built-in
  - format: ---*DD* (note the 3 dashes)
  - format: --*MM*--
  - a hex string
  - a base64 string
  - **http://www.xfront.com**
  - a namespace qualified name
  - a NOTATION from the XML spec

Built-in τύποι δεδομένων (συν.)

Derived types	Subtype of primitive datatype
negativeInteger	negative infinity to -1
long	**-9223372036854775808** to **9223372036854775807**
int	**-2147483648** to **2147483647**
short	**-32768** to **32767**
byte	**-127** to **128**
nonNegativeInteger	0 to infinity
unsignedLong	**0** to **18446744073709551615**
unsignedInt	**0** to **4294967295**
unsignedShort	**0** to **65535**
unsignedByte	**0** to **255**
positiveInteger	**1** to infinity

Σημ: Οι ακόλουθοι τύποι μπορούν να χρησιμοποιηθούν με attributes (which we will discuss later): ID, IDREF, IDREFS, NMTOKEN, NMTOKENS, ENTITY, and ENTITIES.

Δημιουργώντας νέους τύπους

- Ενας νέος τύπος δεδομένων ορίζεται από έναν υπάρχων (ονομαζόμενος "base" type) ορίζοντας τιμές για ένα ή περισσότερα από τα *facets* του base type
- Παράδειγμα. Ο τύπος string έχει 6 προαιρετικά optional facets:
  - length
  - minLength
  - maxLength
  - pattern
  - enumeration
  - whitespace (επιτρεπτές τιμές: preserve, replace, collapse)

Παράδειγμα δημιουργίας νέου τύπου ορίζοντας τιμές στα Facets

```
<xsd:simpleType name="TelephoneNumber">①
   <xsd:restriction base="xsd:string">②
      <xsd:length value="8"/>③
      <xsd:pattern value="\d{3}-\d{4}"/>④
   </xsd:restriction>
</xsd:simpleType>
```

1. Δημιουργία νέου τύπου με το όνομα 'TelephoneNumber'.
2. Elements αυτού του τύπου έχουν τιμές string,
3. Αλλά το μήκος του string πρέπει να είναι ακριβώς 8 χαρακτήρες *και*
4. το string πρέπει να ακολουθεί το μοτίβο: ddd-dddd, όπου 'd' είναι ένα 'digit'.

Facets για integer τύπο δεδομένων

- Ο τύπος δεδομένων integer έχει 8 προαιρετικά facets:
  - totalDigits
  - pattern
  - whitespace
  - enumeration
  - maxInclusive
  - maxExclusive
  - minInclusive
  - minExclusive

Παράδειγμα

```
<xsd:simpleType name= "EarthSurfaceElevation">
   <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="-1290"/>
      <xsd:maxInclusive value="29035"/>
   </xsd:restriction>
</xsd:simpleType>
```

Γενική φόρμα δημιουργίας νέου τύπου δεδομένων ορίζοντας τιμές στα Facets

```
<xsd:simpleType name= "name">
   <xsd:restriction base= "xsd:source">
      <xsd:facet value= "value"/>
      <xsd:facet value= "value"/>
      …
   </xsd:restriction>
</xsd:simpleType>
```

Facets:
- length
- minlength
- maxlength
- pattern
- enumeration
- minInclusive
- maxInclusive
- minExclusive
- maxExclusive

...

Sources:
- string
- boolean
- number
- float
- double
- duration
- dateTime
- time

...

Πολλαπλά Facets – Ενώνοντας τα με "*and*", ή "*or*"?

```
<xsd:simpleType name="TelephoneNumber">
   <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
      <xsd:pattern value="\d{3}-\d{4}"/>
   </xsd:restriction>
</xsd:simpleType>
```

Ενα element δηλωμένο τύπουTelephoneNumber
Πρέπει να είναι string με length=8 *και* το string
Πρέπει να ακολουθεί το μοτίβο: 3 digits, παύλα, 4 digits.

```
<xsd:simpleType name="shape">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="circle"/>
      <xsd:enumeration value="triangle"/>
      <xsd:enumeration value="square"/>
   </xsd:restriction>
</xsd:simpleType>
```

Ενα element δηλωμένο τύπου shape
Πρέπει να είναι string με τιμή circle, *ή* triangle, *ή* square.

Patterns, enumerations => "or" them together
All other facets => "and" them together

Element που περιέχει ένα User-Defined Simple Type

Παράδειγμα:

```
<elevation>5240</elevation>
```

Ενας τρόπος δήλωσης του elevation element:

```
<xsd:simpleType name="EarthSurfaceElevation">
   <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="-1290"/>
      <xsd:maxInclusive value="29035"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:element name="elevation" type="EarthSurfaceElevation"/>
```

Element που περιέχει ένα User-Defined Simple Type (συν)

Ενας εναλακτικός τρόπος δήλωσης του elevation:

```
<xsd:element name="elevation">
   <xsd:simpleType>
      <xsd:restriction base="xsd:integer">
         <xsd:minInclusive value="-1290"/>
         <xsd:maxInclusive value="29035"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

Ο ορισμός simpleType ορίζεται inline, ανώνυμα.

Το μειονέκτημα αυτής της Προσέγγισης είναι ότι αυτό το simpleType δεν μπορεί να επαναχρησιμοποιηθεί από άλλα elements.

Ανακεφαλαίωση δήλωσης Elements (3 τρόποι)

1

```
<xsd:element name="name" type="type" minOccurs="int" maxOccurs="int"/>
```

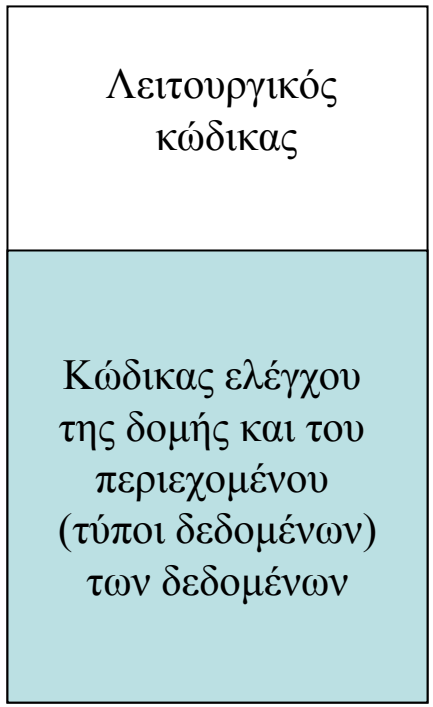
2

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
    <xsd:complexType>
        …
    </xsd:complexType>
</xsd:element>
```

3

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
    <xsd:simpleType>
        <xsd:restriction base="type">
        …
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

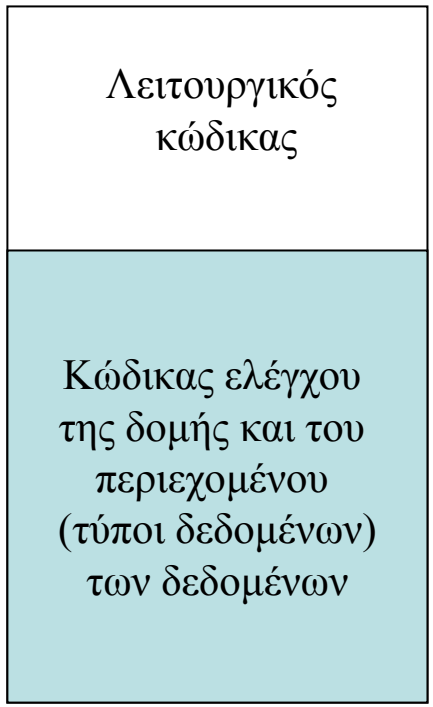
Γλυτώνοντας $$$ χρησιμοποιώντας XML Schemas



«σε ένα τυπικό πρόγραμμα, μέχρι και 60% του κώδικα σπαταλιέται ελέγχοντας τα δεδομένα!"

- source unknown

Γλυτώνοντας $$$ χρησιμοποιώντας XML Schemas (συν)



Αν τα δεδομένα είναι δομημένα σε XML, και υπάρχει schema,
Μπορείς να αναθέσεις τον έλεγχο των Δεδομένων σε έναν
schema validator.

Ο κώδικας μειώνεται ως και 60%!!!

Μεγάλη οικονομία $$!

Regular Expressions

- Ο τύπος δεδομένων string έχει ένα facet με το όνομα pattern. Η τιμή του pattern είναι μια κανονική έκφραση - regular expression. Παρακάτω παρατίθενται παραδείγματα:

Regular Expression	Παράδειγμα
- Chapter \d	- Chapter 1
- Chapter \d	- Chapter 1
- a*b	- b, ab, aab, aaab, …
- [xyz]b	- xb, yb, zb
- a?b	- b, ab
- a+b	- ab, aab, aaab, …
- [a-c]x	- ax, bx, cx

Regular Expressions (συν.)

- Regular Expression
  - [a-c]x
  - [-ac]x
  - [ac-]x
  - [^0-9]x
  - \Dx
  - Chapter\s\d
  - (ho){2} there
  - (ho\s){2} there
  - .abc
  - (a|b)+x

- Παράδειγμα
  - ax, bx, cx
  - -x, ax, cx
  - ax, cx, -x
  - any non-digit char followed by x
  - any non-digit char followed by x
  - Chapter followed by a blank followed by a digit
  - hoho there
  - ho ho there
  - any (*one*) char followed by abc
  - ax, bx, aax, bbx, abx, bax,...

Regular Expressions (σuv.)

- a{1,3}x
- a{2,}x
- \w\s\w

- ax, aax, aaax
- aax, aaax, aaaax, …
  - word **character** (alphanumeric plus dash) followed by a space followed by a word character

- [a-zA-Z-[OI]]*

- A string comprised of any lower and upper case letters, except "O" and "I"

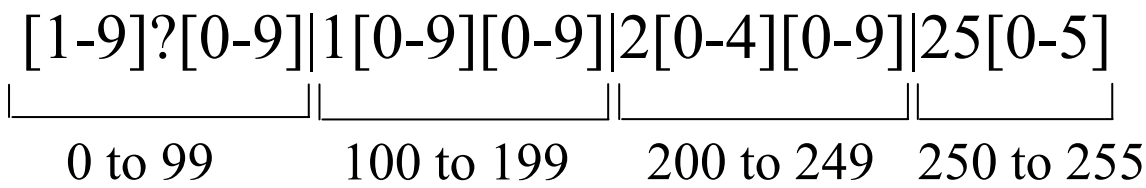
- \.

- The period "." (Without the backward slash the period means "any character")

Regular Expressions (συν.)

\n	linefeed
\r	carriage return
\t	tab
\\	The backward slash \
|	The vertical bar |
\-	The hyphen -
\^	The caret ^
\?	The question mark ?
*	The asterisk *
\+	The plus sign +
\{	The open curly brace {
\}	The close curly brace }
\(	The open paren (
\)	The close paren)
\[	The open square bracket [
\]	The close square bracket]

Παράδειγμα R.E.



Η παραπάνω regular expression περιορίζει ένα *string*
Να έχει τιμές από 0 ως 255.
… χρήσιμη για περιγραφή IP address ...

Derived Types

- Μπορούμε να επιτύχουμε subclassing complexType ορισμών. Αυτό ονομάζεται "derived types"
  - derive by extension: επέκταση του complexType με περισσότερα elements
  - derive by restriction: δημιουργία ενός τύπου που είναι υποσύνολο του base type. 2 τρόποι:
    - Επαναπροσδιορισμός ενός base type element να δέχεται **περιορισμένο εύρος τιμών**, ή
    - Επαναπροσδιορισμός ενός base type element να δέχεται ένα **περιορισμένο αριθμό στιγμιότυπων**.

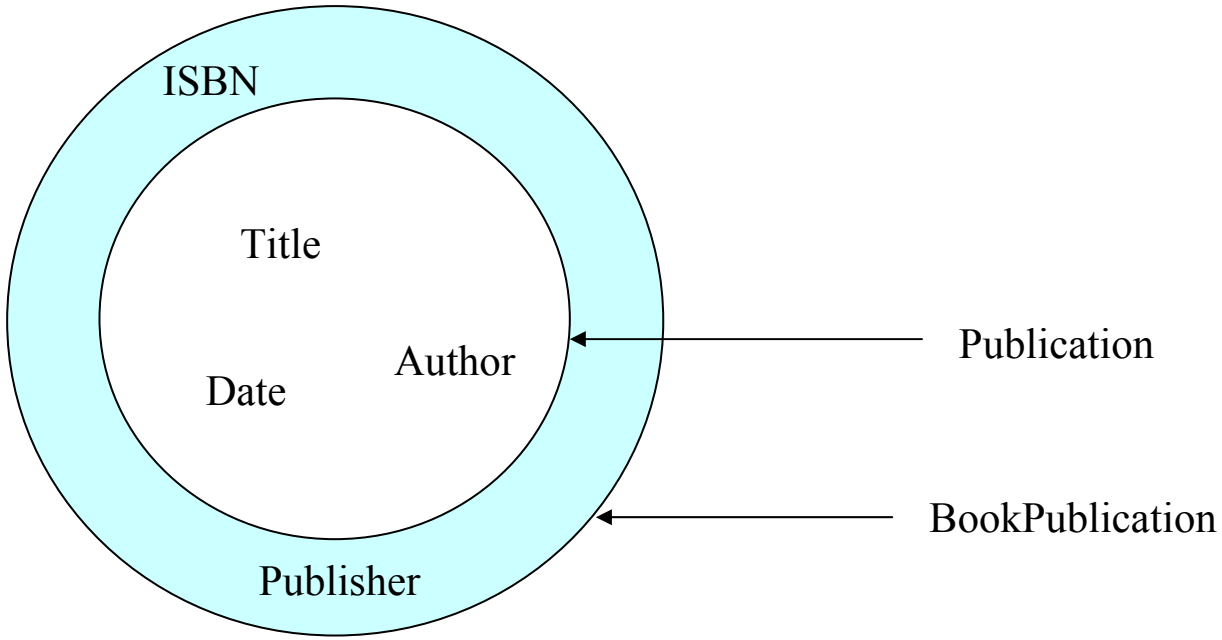
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
               elementFormDefault="qualified">
  <xsd:complexType name="Publication">
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:gYear"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="BookPublication">
    <xsd:complexContent>
      <xsd:extension base="Publication" >
        <xsd:sequence>
          <xsd:element name="ISBN" type="xsd:string"/>
          <xsd:element name="Publisher" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" type="BookPublication" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

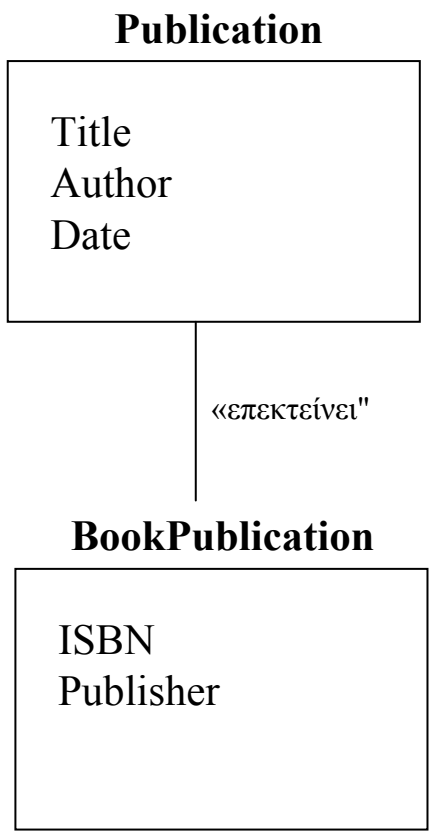
Ο BookPublication επεκτείνει Τον τύπο Publication
Δηλ: Derive by Extension

(see example06)

```
<xsd:complexType name="Publication">
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:gYear"/>
    </xsd:sequence>
</xsd:complexType >
<xsd:complexType name="BookPublication">
   <xsd:complexContent>
      <xsd:extension base="Publication">
         <xsd:sequence>
            <xsd:element name="ISBN" type="xsd:string"/>
            <xsd:element name="Publisher" type="xsd:string"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType >
```

Elements δηλωμένα να είναι τύπου BookPublication θα έχουν 5 child elements - Title, Author, Date, ISBN, και Publisher. Τα elements του derived type προστίθενται στα elements του base type.


ISBN
Title
Author
Date
Publisher
Publication
BookPublication


Publication
Title
Author
Date
«επεκτείνει"
BookPublication
ISBN
Publisher

Derive by Restriction

```
<xsd:complexType name="Publication">
      <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Date" type="xsd:gYear"/>
      </xsd:sequence>
</xsd:complexType>
<xsd:complexType name= "SingleAuthorPublication">
   <xsd:complexContent>
      <xsd:restriction base="Publication">
         <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Author" type="xsd:string"/>
            <xsd:element name="Date" type="xsd:gYear"/>
         </xsd:sequence>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
```

Elements του τύπου SingleAuthorPublication θα έχουν 3 child elements - Title, Author, και Date. Πρέπει να υπάρχει ακριβώς Author element.

Στον τύπο restriction πρέπει να επαναλαμβάνονται όλες οι δηλώσεις του base type (εκτός όταν ο base type έχει element με minOccurs="0" και ο subtype θέλει να το σβήσει -→).

Διαγράφοντας ένα element του base type

```
<xsd:complexType name="Publication">
      <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Author" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Date" type="xsd:gYear"/>
      </xsd:sequence>
</xsd:complexType>
<xsd:complexType name= "ZeroAuthorPublication">
   <xsd:complexContent>
      <xsd:restriction base="Publication">
         <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Date" type="xsd:gYear"/>
         </xsd:sequence>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
```

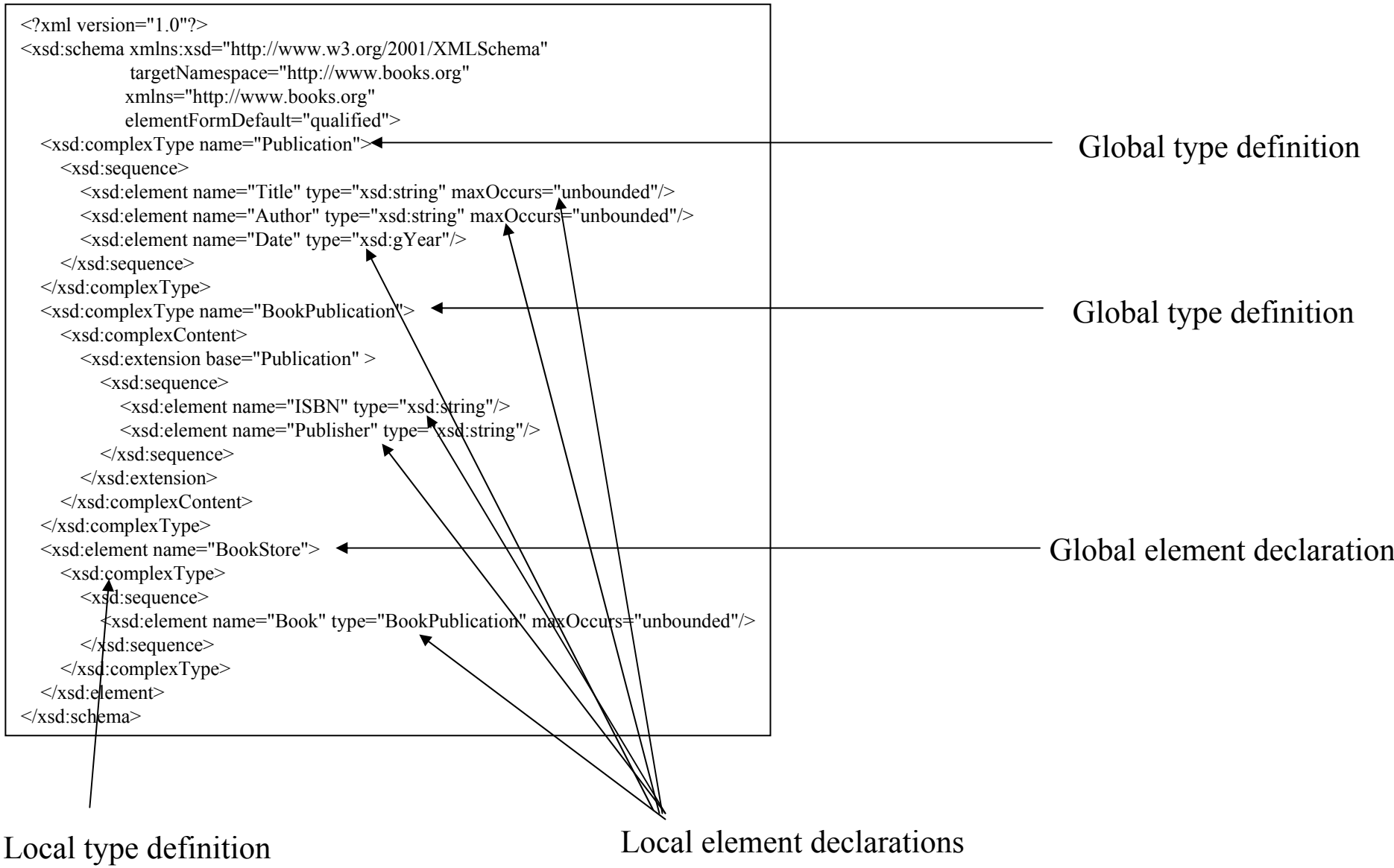
Ορολογία: Δήλωση vs Ορισμός

- Σε ένα schema:
  - *Δηλώνεις* elements και attributes. Συστατικά του Schema που *δηλώνονται* είναι αυτά που αναπαρίστανται στο XML instance document.
  - *Ορίζεις* συστατικά που χρησιμοποιούνται μέσα στο schema. Συστατικά του Schema που *ορίζονται* είναι αυτά που δεν αναπαρίστανται στο XML instance document.

δηλώσεις:	**ορισμοί**:
- element declarations - attribute declarations	- type (simple, complex) definitions - attribute group definitions - model group definitions

Ορολογία: Global vs Local

- Global δηλώσεις element, global ορισμοί τύπων:
  - Είναι δηλώσεις element/ ορισμοί τύπων που είναι άμεσα παιδιά του <schema>
- Local δηλώσεις element, local ορισμοί τύπων :
  - Είναι δηλώσεις element/ ορισμοί τύπων που ενθυλακώνονται μέσα σε άλλα elements/ τύπους.


<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.books.org"
xmlns="http://www.books.org"
elementFormDefault="qualified">
<xsd:complexType name="Publication">
<xsd:sequence>
<xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
<xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
<xsd:element name="Date" type="xsd:gYear"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BookPublication">
<xsd:complexContent>
<xsd:extension base="Publication" >
<xsd:sequence>
<xsd:element name="ISBN" type="xsd:string"/>
<xsd:element name="Publisher" type="xsd:string"/>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
<xsd:element name="BookStore">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="Book" type="BookPublication" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
Global type definition
Global type definition
Global element declaration
Local type definition
Local element declarations

Global vs Local … και τι έγινε?

- Και τί έγινε αν ένα element ή τύπος είναι global ή local. Πρακτικά, τί σημαίνει?
  - Απάντηση: **μόνο global elements/τύποι μπορούν να αναφερθούν (δηλ., επαναχρησιμοποιηθούν)**. Ετσι, αν ένα element/τύπος είναι local τότε είναι ουσιαστικά αόρατο στο υπόλοιπο schema (και σε άλλα schemas).

Attributes

- Ας δούμε μια έκδοση του BookStore DTD με attributes. Μετά, θα δούμε πώς μεταφράζεται σε XML Schemas

```
<!ELEMENT BookStore (Book+)>
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>
<!ATTLIST Book
        Category (autobiography | non-fiction | fiction) #REQUIRED
        InStock (true | false) "false"
        Reviewer CDATA " ">
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

BookStore.dtd

```
<xsd:element name="BookStore">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Book"  maxOccurs="unbounded">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="Title" type="xsd:string"/>
                  <xsd:element name="Author" type="xsd:string"/>
                  <xsd:element name="Date" type="xsd:string"/>
                  <xsd:element name="ISBN" type="xsd:string"/>
                  <xsd:element name="Publisher" type="xsd:string"/>
               </xsd:sequence>
               <xsd:attributeGroup ref="BookAttributes"/>
            </xsd:complexType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
<xsd:attributeGroup name="BookAttributes">
   <xsd:attribute name="Category" use="required">
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="autobiography"/>
            <xsd:enumeration value="non-fiction"/>
            <xsd:enumeration value="fiction"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:attribute>
   <xsd:attribute name="InStock" type="xsd:boolean" default="false"/>
   <xsd:attribute name="Reviewer" type="xsd:string" default=" "/>
</xsd:attributeGroup>
```

→ Category (autobiography | non-fiction | fiction) #REQUIRED

→ InStock (true | false) "false"

→ Reviewer CDATA " "

(see example07)

```
<xsd:attribute name="Category"  use="required">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="autobiography"/>
                <xsd:enumeration value="non-fiction"/>
                <xsd:enumeration value="fiction"/>
            </xsd:restriction>
        </xsd:simpleType>
</xsd:attribute>
```

"Instance documents υποχρεούνται να έχουν το Category attribute (όπως δηλώνεται από τη χρήση "required"). Η τιμή του Category πρέπει να είναι ή autobiography, ή non-fiction, ή fiction (όπως ορίζεται από τα enumeration facets)."

Σημ: Τα attributes μπορούν να έχουν μόνο simpleTypes (δηλ., τα attributes δεν μπορούν να έχουν child elements).

Ανακεφαλαίωση δήλωσης Attributes (2 τρόποι)

1

<xsd:attribute name="*name*" type="*simple-type*" use="*how-its-used*" default/fixed="*value*"/>

xsd:string
xsd:integer
xsd:boolean
...

required
optional
prohibited

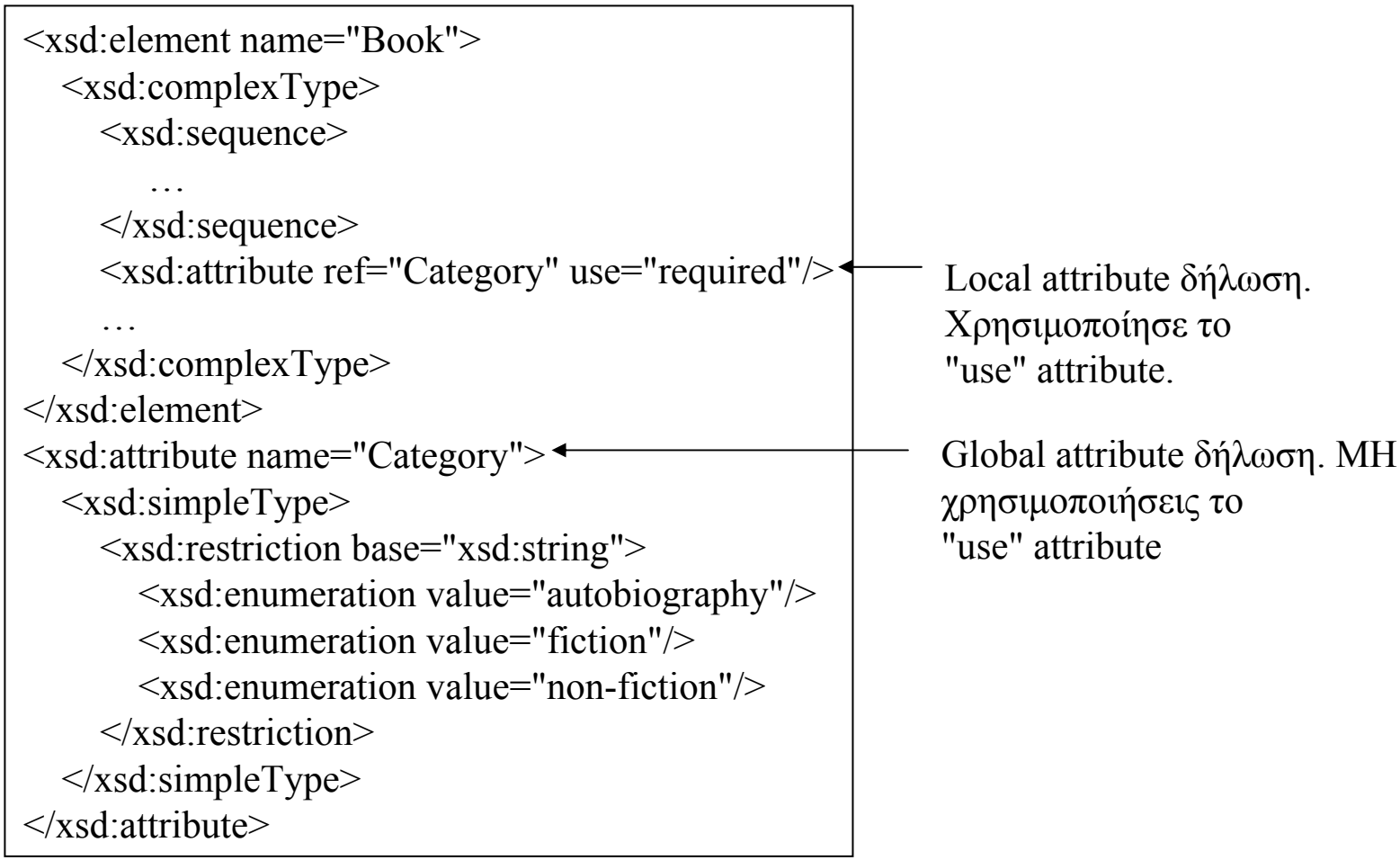
Το "use" attribute πρέπει να είναι optional όταν χρησιμοποιείται default ή fixed.

2

```
<xsd:attribute name="name" use="how-its-used" default/fixed="value">
    <xsd:simpleType>
        <xsd:restriction base="simple-type">
            <xsd:facet value="value"/>
            …
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
```

use --> χρησιμοποίησέ το μόνο σε δηλώσεις Local Attributes

- Το "use" attribute έχει νόημα μόνο σε μια δήλωση element. Για παράδειγμα: "for each Book element, the Category attribute is required".
- Οταν δηλώνεις ένα global attribute μη χρησιμοποιείς το "use"


<xsd:element name="Book">
<xsd:complexType>
<xsd:sequence>
…
</xsd:sequence>
<xsd:attribute ref="Category" use="required"/>
…
</xsd:complexType>
</xsd:element>
<xsd:attribute name="Category">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:enumeration value="autobiography"/>
<xsd:enumeration value="fiction"/>
<xsd:enumeration value="non-fiction"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
Local attribute δήλωση. Χρησιμοποίησε το "use" attribute.
Global attribute δήλωση. ΜΗ χρησιμοποιήσεις το "use" attribute

Inlining Attributes

- Στην επόμενη διαφάνεια δείχνεται άλλος ένας τρόπος δήλωσης attribute - Τα attributes are inlined μέσα στη δήλωση του Book

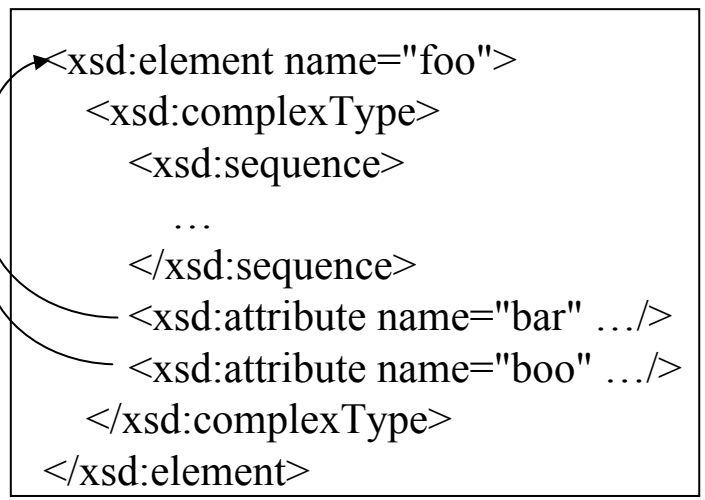
```
<xsd:element name="Book"  maxOccurs="unbounded">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Title" type="xsd:string"/>
            <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Date" type="xsd:string"/>
            <xsd:element name="ISBN" type="xsd:string"/>
            <xsd:element name="Publisher" type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="Category" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="autobiography"/>
                    <xsd:enumeration value="non-fiction"/>
                    <xsd:enumeration value="fiction"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="InStock" type="xsd:boolean" default="false"/>
        <xsd:attribute name="Reviewer" type="xsd:string" default=" "/>
    </xsd:complexType>
</xsd:element>
```

(see example08)

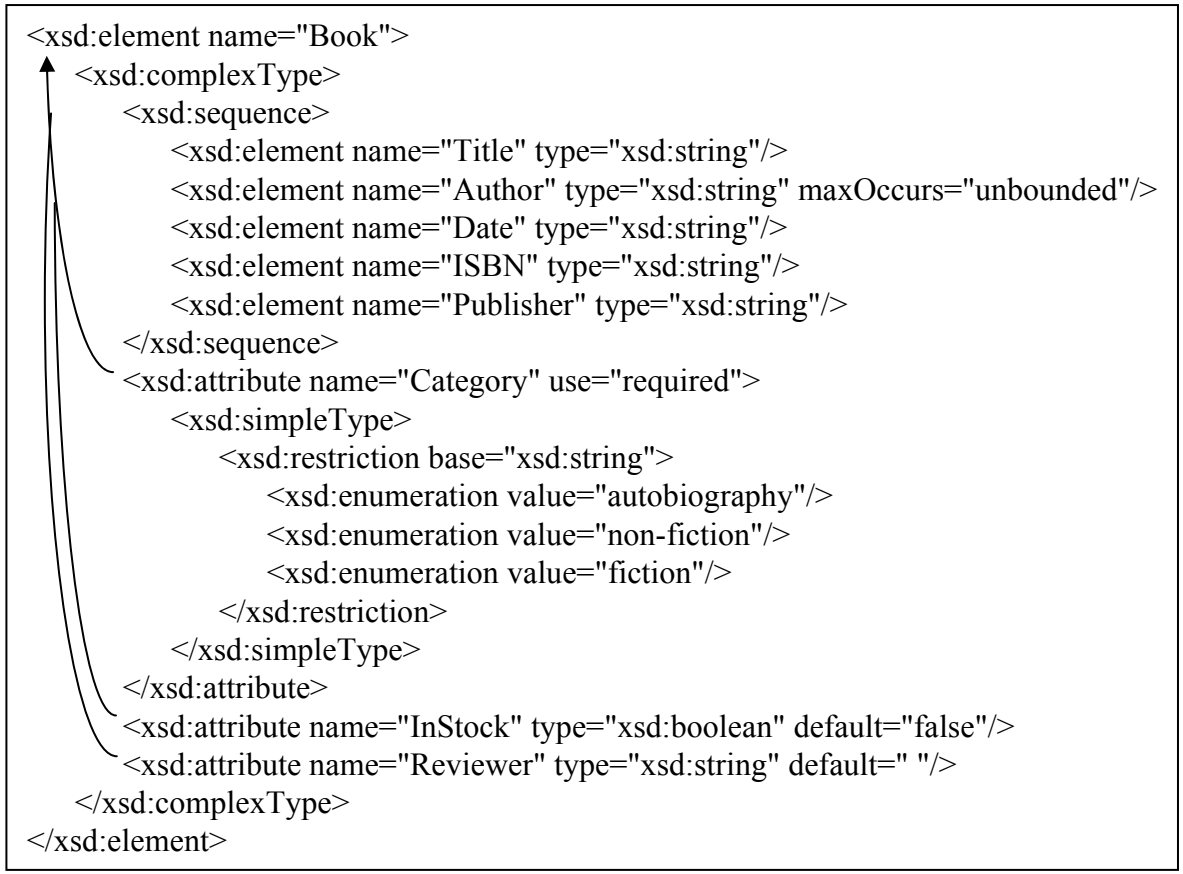
Σημειώσεις για Attributes

- Οι δηλώσεις των attributes γίνονται στο τέλος, μετά τις δηλώσεις των elements.
- *Τα attributes είναι πάντα σχετικά με το element στο οποίο ορίζονται (nested).*

"bar και boo είναι attributes του foo"



Τα attributes ανήκουν στο element στο οποίο ενθυλακώνονται (Book) Δηλ, το Book έχει 3 attributes - Category, InStock, και Reviewer.



```
<xsd:element name="Book">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Title" type="xsd:string"/>
            <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Date" type="xsd:string"/>
            <xsd:element name="ISBN" type="xsd:string"/>
            <xsd:element name="Publisher" type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="Category" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="autobiography"/>
                    <xsd:enumeration value="non-fiction"/>
                    <xsd:enumeration value="fiction"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="InStock" type="xsd:boolean" default="false"/>
        <xsd:attribute name="Reviewer" type="xsd:string" default=" "/>
    </xsd:complexType>
</xsd:element>
```

Element με απλό περιεχόμενο και Attributes

Παράδειγμα. Θεωρείστε το ακόλουθο:

```
<elevation units="feet">5440</elevation>
```

Το elevation element έχει 2 περιορισμούς:

- Εχει απλό (integer) περιεχόμενο
- Εχει 1 attribute, το units

Πώς δηλώνουμε το elevation? →

```
<xsd:element name="elevation">
   <xsd:complexType> (1)
      <xsd:simpleContent> (2)
         <xsd:extension base="xsd:integer"> (3)
            <xsd:attribute name="units" type="xsd:string" use="required"/> (4)
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

1. Το elevation περιέχει ένα attribute.
   - άρα, πρέπει να χρησιμοποιήσουμε <xsd:complexType>
2. Ομως, το elevation δεν περιέχει child elements (για το οποίο χρησιμοποιούμε το <complexType>). Αντίθετα, το elevation περιέχει simpleContent.
3. Επεκτείνουμε το simpleContent (integer) ...
4. με ένα attribute.

Ανακεφαλαίωση δήλωσης Elements

1. **Element με απλό περιεχόμενο - Simple Content.**

Δηλώνουμε το element χρησιμοποιώντας έναν **built-in τύπο**:

```
<xsd:element name="numStudents" type="xsd:positiveInteger"/>
```

Δηλώνουμε το element χρησιμοποιώντας έναν **user-defined simpleType**:

```
<xsd:simpleType name="shapes">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="triangle"/>
      <xsd:enumeration value="rectangle"/>
      <xsd:enumeration value="square"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:element name="geometry" type="shapes"/>
```

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε **inline simpleType** ορισμό:

```
<xsd:element name="geometry">
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="triangle"/>
         <xsd:enumeration value="rectangle"/>
         <xsd:enumeration value="square"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

Ανακεφαλαίωση δήλωσης Elements

2.Το **Element περιέχει Child Elements**

Ορίζοντας child elements **inline**:

```
<xsd:element name="Person">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Title" type="xsd:string"/>
         <xsd:element name="FirstName" type="xsd:string"/>
         <xsd:element name="Surname" type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Εναλλακτικά, μπορούμε να δημιουργήσουμε ένα **named complexType** τον οποίο και να χρησιμοποιήσουμε:

```
<xsd:complexType name="PersonType">
    <xsd:sequence>
       <xsd:element name="Title" type="xsd:string"/>
       <xsd:element name="FirstName" type="xsd:string"/>
       <xsd:element name="Surname" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name="Person" type="PersonType"/>
```

Ανακεφαλαίωση δήλωσης Elements

3. Το **Element περιέχει έναν complexType που είναι επέκταση άλλου complexType**

```
<xsd:complexType name="Publication">
   <xsd:sequence>
      <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:gYear"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BookPublication">
   <xsd:complexContent>
      <xsd:extension base="Publication" >
         <xsd:sequence>
            <xsd:element name="ISBN" type="xsd:string"/>
            <xsd:element name="Publisher" type="xsd:string"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Book" type="BookPublication"/>
```

Ανακεφαλαίωση δήλωσης Elements

4. Το **Element περιέχει έναν complexType που είναι περιορισμός άλλου complexType**

```
<xsd:complexType name="Publication">
      <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Date" type="xsd:gYear"/>
      </xsd:sequence>
</xsd:complexType>
<xsd:complexType name= "SingleAuthorPublication">
   <xsd:complexContent>
      <xsd:restriction base="Publication">
         <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Author" type="xsd:string"/>
            <xsd:element name="Date" type="xsd:gYear"/>
         </xsd:sequence>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Catalogue" type="SingleAuthorPublication"/>
```

Ανακεφαλαίωση δήλωσης Elements

5. Το **Element περιέχει απλό περιεχόμενο - Simple Content και Attributes**

```
<xsd:element name="apple">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="variety" type="xsd:string" use="required"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

Παράδειγμα. <apple variety="Cortland">Large, green, sour</apple>

complexContent vs simpleContent

- Με complexContent επεκτείνεις ή περιορίζεις complexType
- Με simpleContent επεκτείνεις ή περιορίζεις simpleType

```
<xsd:complexType name="…">
   <xsd:complexContent>
      <extension base="X">
         …
      </extension>
   </xsd:complexContent>
</xsd:complexType>
```

vs

```
<xsd:complexType name="…">
   <xsd:simpleContent>
      <extension base="Y">
         …
      </extension>
   </xsd:simpleContent>
</xsd:complexType>
```

X πρέπει να είναι complexType

Y πρέπει να είναι simpleType

group Element

- Το group element δίνει τη δυνατότητα ομαδοποίησης δηλώσεων elements.
- Σημ: Το group element δεν επιτρέπει την ομαδοποίηση δηλώσεων attributes!

```
<xsd:element name="Book" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:group ref="PublicationElements"/>
            <xsd:element name="ISBN" type="string"/>
            <xsd:element name="Reviewer" type="string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="CD" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:group ref="PublicationElements"/>
            <xsd:element name="RecordingStudio" type="string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:group name="PublicationElements">
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
      <xsd:element name="Date" type="xsd:string"/>
    </xsd:sequence>
</xsd:group>
```

Παράδειγμα χρήσης του <group> element

Σημειώσεις για το group

- Οι ορισμοί Group πρέπει να είναι global

```
<xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:group name="PublicationElements">
          <xsd:sequence>
            <xsd:element name="Title" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Author" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="Date" type="xsd:string"/>
          </xsd:sequence>
        </xsd:group>
        <xsd:element name="ISBN" type="xsd:string"/>
        <xsd:element name="Publisher" type="xsd:string"/>
      </xsd:sequence>
      ...
    </xsd:complexType>
  </xsd:element>
```

Δε επιτρέπεται inline τον ορισμό group.
Πρέπει να χρησιμοποιήσεις Ένα *ref* και να ορίσεις το group globally.

Εκφράζοντας επιλογές

DTD:

```
<!ELEMENT transportation (train | plane | automobile)>
```

XML Schema:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.travel.org"
            xmlns="http://www.travel.org"
            elementFormDefault="qualified">
  <xsd:element name="transportation">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element name="train" type="xsd:string"/>
        <xsd:element name="plane" type="xsd:string"/>
        <xsd:element name="automobile" type="xsd:string"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

(see example10)

Σημ: Η επιλογή – choice είναι exclusive-or, δηλ, transportation μπορεί να περιέχει μόνο **ένα** element - train, ή plane, ή automobile.

Εκφράζοντας επαναλαμβανόμενες επιλογές

DTD:

```
<!ELEMENT binary-string (zero | one)*>
```

XML Schema:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.binary.org"
            xmlns="http://www.binary.org"
            elementFormDefault="qualified">
  <xsd:element name="binary-string">
    <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="zero" type="xsd:unsignedByte" fixed="0"/>
        <xsd:element name="one" type="xsd:unsignedByte" fixed="1"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

(see example 11)

Σημ:

1. Ενα element μπορεί να φτιάξει - fix την τιμή του, με το fixed attribute.

fixed/default τιμές Element

- Οταν δηλώνεις ένα element, μπορείς να του δώσεις μια fixed ή default τιμή
  - Τότε, στο instance document, μπορείς να αφήσεις το element άδειο

```
<element name="zero" fixed="0"/>
…
<zero>0</zero>

    ή ομοίως:

<zero/>
```

```
<element name="color" default="red"/>
…
<color>red</color>

 ή ομοίως:

<color/>
```

Χρησιμοποιώντας <sequence> και <choice>

DTD:

```
<!ELEMENT life ((work, eat)*, (work | play), sleep)* >
```

XML Schema:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.life.org"
                xmlns="http://www.life.org"
                elementFormDefault="qualified">
  <xsd:element name="life">
    <xsd:complexType>
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="work" type="xsd:string"/>
          <xsd:element name="eat" type="xsd:string"/>
        </xsd: sequence>
        <xsd:choice>
          <xsd:element name="work" type="xsd:string"/>
          <xsd:element name="play" type="xsd:string"/>
        </xsd:choice>
        <xsd:element name="sleep" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Εκφράζοντας οποιαδήποτε σειρά εμφάνισης - Any Order

Πρόβλημα: δημιουργία του element Book, που περιέχει Author, Title, Date, ISBN, και Publisher, *σε any order* (Σημ: αυτό είναι δύσκολο και άσχημο στα DTDs).

XML Schema:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book"  maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:all>
              <xsd:element name="Title" type="xsd:string"/>
              <xsd:element name="Author" type="xsd:string"/>
              <xsd:element name="Date" type="xsd:string"/>
              <xsd:element name="ISBN" type="xsd:string"/>
              <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:all>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

(see example 12)

<all> σημαίνει ότι το Book πρέπει να περιέχει όλα τα child elements, σε οποιαδήποτε σειρά.

Περιορισμοί χρήσης του <all>

- Elements δηλωμένα μέσα σε <all> πρέπει να έχουν τιμή maxOccurs ίση με "1" (minOccurs μπορεί να είναι "0" ή "1")
- Αν ένας complexType χρησιμοποιεί <all> και επεκτείνει κάποιον άλλο τύπο, τότε ο τύπος πατέρας πρέπει να έχει άδειο περιεχόμενο.
- Το <all> δεν μπορεί να ενθυλακωθεί σε <sequence>, <choice>, ή άλλο <all>
- Τα περιεχόμενα του <all> πρέπει να είναι απλά elements. Δεν μπορεί να περιέχει <sequence> ή <choice>

Empty Element

DTD:

```
<!ELEMENT image EMPTY>
<!ATTLIST image href CDATA #REQUIRED>
```

Schema:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.photography.org"
            xmlns="http://www.photography.org"
            elementFormDefault="qualified">
  <xsd:element name="gallery">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="image" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:attribute name="href" type="xsd:anyURI" use="required"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

(see example 13)

Instance doc (snippet):

```
<image href="http://www.xfront.com/InSubway.gif"/>
```

No targetNamespace (noNamespaceSchemaLocation)

- Μερικές φορές θέλεις να δημιουργήσεις ένα schema χωρίς να συσχετίσεις τα elements με ένα namespace
- Το targetNamespace attribute είναι ένα προαιρετικό attribute του <schema>. Ετσι, αν δε θέλεις να δηλώσεις namespace για το schema σου, μη χρησιμοποιήσεις το targetNamespace attribute.
- Συνέπειες μή ύπαρξης namespace
  - 1. Στο instance document δεν εξειδικεύεις τα elements.
  - 2. Στο instance document, αντί να χρησιμοποιείς schemaLocation χρησιμοποίησε noNamespaceSchemaLocation.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">
   <xsd:element name="BookStore">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="Book" minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="Book">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="Title"/>
            <xsd:element ref="Author"/>
            <xsd:element ref="Date"/>
            <xsd:element ref="ISBN"/>
            <xsd:element ref="Publisher"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="Title" type="xsd:string"/>
   <xsd:element name="Author" type="xsd:string"/>
   <xsd:element name="Date" type="xsd:string"/>
   <xsd:element name="ISBN" type="xsd:string"/>
   <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Δεν υπάρχει targetNamespace attribute, δεν υπάρχει default namespace.

(see example14)

```
<?xml version="1.0"?>
<BookStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation= "BookStore.xsd">
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>1998</Date>
        <ISBN>1-56592-235-2</ISBN>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
    …
</BookStore>
```

(see example14)

1. Δεν υπάρχει default δήλωση namespace. Οπότε, κανένα από τα elements δεν σχετίζεται με namespace.
2. Δεν χρησιμοποιείται xsi:schemaLocation. Αντίθετα, χρησιμοποιείται xsi:noNamespaceSchemaLocation.

Δημιουργώντας λίστες

- Μερικές φορές θέλουμε ένα element να περιέχει λίστα τιμών, π.χ., «Το περιεχόμενο του Numbers element είναι μια λίστα αριθμών».

Παράδειγμα: Για ένα έγγραφο που περιέχει μια λοταρία, έχουμε:

<Numbers>12 49 37 99 20 67</Numbers>

Πώς δηλώνουμε το element Numbers ...

(1) να περιέχει μια λίστα από integers, και
(2) Κάθε integer να περιορίζεται από 1 ως 99, και
(3) ο συνολικός αριθμός στη λίστα να είναι ακριβώς 6.

```
<?xml version="1.0"?>
<LotteryDrawings xmlns="http://www.lottery.org"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation=
                         "http://www.lottery.org
                          Lottery.xsd">
    <Drawing>
        <Week>July 1</Week>
        <Numbers>21 3 67 8 90 12</Numbers>
    </Drawing>
    <Drawing>
        <Week>July 8</Week>
        <Numbers>55 31 4 57 98 22</Numbers>
    </Drawing>
    <Drawing>
        <Week>July 15</Week>
        <Numbers>70 77 19 35 44 11</Numbers>
    </Drawing>
</LotteryDrawings>
```

Lottery.xml (see example19)

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.lottery.org"
                xmlns="http://www.lottery.org"
                elementFormDefault="qualified">
  <xsd:simpleType name="LotteryNumbers">
    <xsd:list itemType="xsd:positiveInteger"/>
  </xsd:simpleType>
  <xsd:element name="LotteryDrawings">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Drawing" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Week" type="xsd:string"/>
              <xsd:element name="Numbers" type="LotteryNumbers"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Lottery.xsd

LotteryNumbers --> Need Stronger Datatyping

- Η λίστα στο προηγούμενο schema είχε 2 προβλήματα:
  - Επέτρεψε στα <Numbers> να περιέχουν αυθαίρετα μεγάλη λίστα
  - Οι αριθμοί στη λίστα μπορεί να είναι οποιοσδήποτε positiveInteger
- Πρέπει να:
  - Να περιορίσουμε το μήκος της λίστας σε τιμή="6"
  - Να περιορίσουμε τα Numbers σε maxInclusive τιμή="99"

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.lottery.org"
                xmlns="http://www.lottery.org"
               elementFormDefault="qualified">
  <xsd:simpleType name="OneToNinetyNine">
    <xsd:restriction base="xsd:positiveInteger">
      <xsd:maxInclusive value="99"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="NumbersList">
    <xsd:list itemType="OneToNinetyNine"/>
  </xsd:simpleType>
  <xsd:simpleType name="LotteryNumbers">
    <xsd:restriction base="NumbersList">
      <xsd:length value="6"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:element name="LotteryDrawings">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Drawing" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Week" type="xsd:string"/>
              <xsd:element name="Numbers" type="LotteryNumbers"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Lottery.xsd (see example19)

```
<xsd:simpleType name="OneToNinetyNine">
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:maxInclusive value="99"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="NumbersList">
  <xsd:list itemType="OneToNinetyNine"/>
</xsd:simpleType>
```

```
<xsd:simpleType name="LotteryNumbers">
  <xsd:restriction base="NumbersList">
    <xsd:length value="6"/>
  </xsd:restriction>
</xsd:simpleType>
```

Ο τύπος NumbersList είναι μια λίστα όπου ο τύπος κάθε αντικειμένου της είναι OneToNinetyNi
Ο τύπος LotteryNumbers περιορίζει τον τύπο NumbersList σε μήκος 6

```
<xsd:simpleType name="OneToNinetyNine">
   <xsd:restriction base="xsd:positiveInteger">
      <xsd:maxInclusive value="99"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="NumbersList">
   <xsd:list itemType="OneToNinetyNine"/>
</xsd:simpleType>
<xsd:simpleType name="LotteryNumbers">
   <xsd:restriction base="NumbersList">
      <xsd:length value="6"/>
   </xsd:restriction>
</xsd:simpleType>
```

Εναλλακτικά,

```
<xsd:simpleType name="LotteryNumbers">
    <xsd:restriction>
       <xsd:simpleType>
          <xsd:list itemType="OneToNinetyNine"/>
       </xsd:simpleType>
       <xsd:length value="6"/>
    </xsd:restriction>
</xsd:simpleType>
```

Σημειώσεις για τον τύπο list

- Δεν μπορεί να έχουμε **list από lists**
  - δηλ., δεν μπορείς να δημιουργήσεις ένα τύπο list από άλλο τύπο list.
- δεν μπορείς να δημιουργήσεις λίστα από complexTypes
  - δηλ., **lists εφαρμόζονται μόνο σε simpleTypes**
- Στο instance document, πρέπει να **διαχωρίσεις κάθε αντικείμενο της λίστας με white space** (blank space, tab, ή carriage return)
- Τα μόνα facets του τύπου list είναι:
  - **length**: use this to specify the **length of the list**
  - **minLength**: use this to specify the **minimum length of the list**
  - **maxLength**: use this to specify the **maximum length of the list**
  - **enumeration**: use this to specify the **values that the list may have**
  - **pattern**: use this to specify **the values that the list may have**

Ανακεφαλαίωση δήλωσης simpleTypes

1. **simpleType που χρησιμοποιεί built-in base type:**

```
<xsd:simpleType name= "EarthSurfaceElevation">
   <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="-1290"/>
      <xsd:maxInclusive value="29035"/>
   </xsd:restriction>
</xsd:simpleType>
```

2. **simpleType που χρησιμοποιεί άλλο simpleType ως base type:**

```
<xsd:simpleType name= "BostonSurfaceElevation">
   <xsd:restriction base="EarthSurfaceElevation">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="120"/>
   </xsd:restriction>
</xsd:simpleType>
```

Summary of Declaring simpleTypes

3. **simpleType που ορίζει τύπο list:**

```
<xsd:simpleType name= "LotteryNumbers">
   <xsd:list itemType="OneToNinetyNine"/>
</xsd:simpleType>
```

where the datatype OneToNinetyNine is declared as:

```
<xsd:simpleType name= "OneToNinetyNine">
   <xsd:restriction base="xsd:nonNegativeInteger">
      <xsd:maxInclusive value="99"/>
   </xsd:restriction>
</xsd:simpleType>
```

4. **Εναλακτικά, χρησιμοποιώντας inlined simpleType:**

```
<xsd:simpleType name= "LotteryNumbers">
   <xsd:list>
      <xsd:simpleType>
         <xsd:restriction base="xsd:nonNegativeInteger">
            <xsd:maxInclusive value="99"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:list>
</xsd:simpleType>
```

any Element

- Το <any> element επιτρέπει στον συγγραφέα του XML εγγράφου να επεκτείνει το έγγραφό του με elements που δεν ορίζονται στο schema.

```
<xsd:element name="Book">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="Title" type="xsd:string"/>
                        <xsd:element name="Author" type="xsd:string"/>
                        <xsd:element name="Date" type="xsd:string"/>
                        <xsd:element name="ISBN" type="xsd:string"/>
                        <xsd:element name="Publisher" type="xsd:string"/>
                        <xsd:any  minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
```

Τώρα ένα XML έγγραφο μπορεί να περιέχει και άλλα elements (μετά το<Publisher>)

anyAttribute

- Το <anyAttribute> element επιτρέπει στον συγγραφέα του XML εγγράφου να επεκτείνει το έγγραφό του με Attributes που δεν ορίζονται στο schema.

```
<xsd:element name="Book">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="Title" type="xsd:string"/>
                        <xsd:element name="Author" type="xsd:string"/>
                        <xsd:element name="Date" type="xsd:string"/>
                        <xsd:element name="ISBN" type="xsd:string"/>
                        <xsd:element name="Publisher" type="xsd:string"/>
                        <xsd:any  minOccurs="0"/>
                </xsd:sequence>
                <xsd:anyAttribute/>
        </xsd:complexType>
</xsd:element>
```

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.repository.org"
                xmlns="http://www.repository.org"
                elementFormDefault="qualified">
   <xsd:element name="Reviewer">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Name">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="First" type="xsd:string"/>
                     <xsd:element name="Last" type="xsd:string"/>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:attribute name="id" type="xsd:ID"/>
</xsd:schema>
```

SchemaRepository.xsd (see example24)

```
<?xml version="1.0"?>
<BookSeller xmlns="http://www.BookRetailers.org"
            xmlns:sr="http://www.repository.org"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation=
                  "http://www.BookRetailers.org
                   BookSeller.xsd
                   http://www.repository.org
                   SchemaRepository.xsd">
    <Book sr:id="P.M.">
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>1998</Date>
        <ISBN>1-56592-235-2</ISBN>
        <Publisher>McMillin Publishing</Publisher>
        <Reviewer xmlns="http://www.repository.org">
          <Name>
            <First>Roger</First>
            <Last>Costello</Last>
          </Name>
        </Reviewer>
    </Book>
    <Book sr:id="R.B.">
        <Title>Illusions: The Adventures of a Reluctant Messiah</Title>
        <Author>Richard Bach</Author>
        <Date>1977</Date>
        <ISBN>0-440-34319-4</ISBN>
        <Publisher>Dell Publishing Co.</Publisher>
    </Book>
</BookSeller>
```

BookStore.xml (see example24)

Version Management

- Το schema element έχει ένα προαιρετικό attribute, το version, το οποίο μπορεί κανείς να χρησιμοποιήσει για να κρατάει την έκδοση του schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified"
                version="1.0">
    ...
</xsd:schema>
```

nil περιεχόμενο

- Μπορείς να δηλώσεις σε ένα schema ότι ένα element μπορεί να είναι nil στο instance document. Empty περιεχόμενο vs nil:
  - Empty: Το element με empty περιεχόμενο περιορίζεται να μην έχει περιεχόμενο.
  - nil: Το instance document element μπορεί να πει ότι δεν υπάρχει διαθέσιμη τιμή θέτοντας το attribute - xsi:nil – να ισούται με 'true'

XML Schema:

```
<xsd:element name="PersonName">
    <xsd:complexType>
        <xsd:element name="forename" type="xsd:NMTOKEN"/>
        <xsd:element name="middle" type="xsd:NMTOKEN" nillable="true"/>
        <xsd:element name="surname" type="xsd:NMTOKEN"/>
    </xsd:complexType>
</xsd:element>
```

XML instance document:

```
<PersonName>
    <forename>John</forename>
    <middle xsi:nil="true"/>
    <surname>Doe</surname>
</PersonName>
```

Το περιεχόμενο του middle Μπορεί να είναι τιμή τύπου NMTOKEN, ή να μην είναι ορισμένο.

Mixed περιεχόμενο

- Σε όλα τα παραδείγματα το περιεχόμενο κάθε ήταν είτε
  - όλα elements, ή
  - Όλα δεδομένα
- Ενα element που περιέχει συνδυασμό elements και (string) δεδομένων καλείται "mixed content".
- Το Mixed content έχει πολλές εφαρμογές. Π.χ, η XSLT χρησιμοποιεί mixed content συχνά σε template rules, π.χ.,

```
<xsl:template match="Book">
    The title of the book is:
        <xsl:value-of select="Title/text()"/>
    The author of the book is:
        <xsl:value-of select="Author/text()"/>
</xsl:template>
```

Δείτε ότι το περιεχόμενο του xsl:template element είναι συνδυασμός string δεδομένων και elements.

προσδιορίζοντας Mixed Content στη δήλωση ενός Element

- Το <complexType> element έχει ένα προαιρετικό attribute, το mixed. By default, mixed="false".
- Για να προσδιορίσεις ότι ένα element μπορεί να έχει mixed content χρησιμοποίησε <complexType **mixed="true"**>

```
<?xml version="1.0"?>
<Letter xmlns="http://www.letter.org"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation=
                    "http://www.letter.org
                     Letter.xsd">
  <Body>
    Dear Sirs:
       This letter is to inform you that we are
    are finding your tool <emp> very </emp> useful.
  </Body>
</Letter>
```

Letter.xml (see example36)

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.letter.org"
                xmlns="http://www.letter.org"
                elementFormDefault="qualified">
   <xsd:element name="Letter">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Body">
               <xsd:complexType mixed="true">
                  <xsd:sequence>
                     <xsd:element name="emp" type="xsd:string"/>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Letter.xsd (see example36)