ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

# ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

# Graphical XML Schema


**Της φοιτήτριας**

**Αικατερίνης-Ιωσηφίνα Αρβανιτάκη**

**Αρ. Μητρώου: 06/3128**

**Επιβλέπων καθηγητής**

**Κεραμόπουλος Ευκλείδης**

Πτυχιακή εργασία της φοιτήτριας Αικατερίνης-Ιωσηφίνα Αρβανιτάκη

**Θεσσαλονίκη 2012**

# ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία πραγματοποιήθηκε στο Αλεξάνδρειο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Θεσσαλονίκης στο τμήμα Πληροφορικής της Σχολής Τεχνολογικών Εφαρμογών(Σ.Τ.ΕΦ).

Στην πτυχιακή μου εργασίας με θέμα graphical xml schemes αρχικά, στο 1ο κεφάλαιο αναλύω το xml schema, την δομή του και τα χαρακτηριστικά του. Στη συνέχεια στο 2ο κεφάλαιο περιγράφω και συγκρίνω διάφορα graphical xml schemes, έπειτα στο 3ο κεφάλαιο αναλύω την τεχνολογία JAXB και τέλος στο 4ο κεφάλαιο υπάρχει μια παρουσίαση της εφαρμογής μου στην πλατφόρμα NetBeans σε συνδυασμό με την JAXB, όπου υπάρχει η δική μου γραφική αναπαράσταση ενός xml schema.

# ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον καθηγητή μου και επιβλέπων καθηγητή της πτυχιακής μου εργασίας κ. Κεραμόπουλο Ευκλείδη για την βοήθεια και υποστήριξη του κατά της διάρκεια εκπόνησης της εργασίας μου.

Επίσης θα ήθελα να ευχαριστήσω τους γονείς μου, για την συμπαράστασή τους όλο αυτό το καιρό.

# Κατάλογος περιεχομένων

Πτυχιακή εργασία της φοιτήτριας Αικατερίνης-Ιωσηφίνα Αρβανιτάκη

# ΚΕΦΑΛΑΙΟ 1

# XML SCHEMA

## ΕΙΣΑΓΩΓΗ

Σήμερα, ζούμε σε έναν κόσμο όπου οι πληροφορίες είναι το παν και όσο πιο γρήγορα έχουμε πρόσβαση σε αυτές, τόσο περισσότερη δύναμη έχουμε. Λαμβάνοντας υπόψη το γεγονός ότι το μόνο μέρος που γνωρίζουμε ότι έχει το μεγαλύτερο ποσό πληροφοριών συγκεντρωμένο στον ίδιο χώρο είναι το Διαδίκτυο, αυτοί που έχουν το "*πάνω χέρι"* σε αυτή την περίπτωση είναι εκείνοι που παίρνουν τις πληροφορίες στο συντομότερο χρονικό διάστημα.

Δεδομένου ότι αυτό είναι αναμφίβολα αλήθεια, φανταστείτε να είχατε ένα εργαλείο που θα σας επέτρεπε να παίρνεται από τα αρχεία τις ακριβείς πληροφορίες που χρειάζεστε και, επίσης, ότι τα αρχεία αυτά είχαν την ίδια δομή σε όλο το Διαδίκτυο. Αυτά τα αρχεία τα οποία μπορούν εύκολα να ανακριθούν είναι τα αρχεία XML και είναι μια τυποποιημένη δόμηση δεδομένων.

XML Schema του W3C προσφέρει ένα ισχυρό σύνολο εργαλείων για τον καθορισμό των αποδεκτών XML εγγράφων δομών και περιεχομένου. Ενώ τα XML σχήματα είναι ισχυρά, αυτή η δύναμη προέρχεται με σημαντική πολυπλοκότητα. Μια ποικιλία από διαφορετικά στυλ για σχήματα γραφής, απλές και σύνθετες μορφές, τύπους δεδομένων και τις απόψεις, τα κλειδιά, την επεκτασιμότητα, την τεκμηρίωση, σχεδιαστικές επιλογές, τις βέλτιστες πρακτικές και τους περιορισμούς.

Πτυχιακή εργασία της φοιτήτριας Αικατερίνης-Ιωσηφίνα Αρβανιτάκη

Τι είναι ένα XML Schema?

Ο σκοπός ενός σχήματος είναι να καθορίσει τα δομικά στοιχεία ενός εγγράφου XML, όπως ακριβώς και ένα DTD.

Ένα XML Schema:

- Προσδιορίζει τα στοιχεία που μπορεί να εμφανιστούν σε ένα έγγραφο.
- Προσδιορίζει τα χαρακτηριστικά που μπορεί να εμφανιστούν σε ένα έγγραφο.
- Καθορίζει ποια στοιχεία είναι στοιχεία-παιδιά.
- Καθορίζει τη σειρά που θα έχουν τα στοιχεία-παιδιά.
- Ορίζει τον αριθμό από στοιχεία-παιδιά.
- Καθορίζει αν ένα στοιχείο είναι άδειο ή μπορεί να περιλαμβάνει κείμενο.
- Ορίζει τους τύπους των δεδομένων.

Συστατικά δομής του σχήματος XML (XML Schema : Structures)

Τα συστατικά δομής του σχήματος XML παρέχουν λειτουργίες για την περιγραφή της δομής και θέτουν περιορισμούς στο περιεχόμενο των XML εγγράφων.
Ένα σχήμα XML αποτελείται από ένα σετ δομικών συστατικών που μπορούν να χωριστούν σε τρείς υποκατηγορίες.

Τα βασικά συστατικά:

Α) Το σχήμα ορισμών και δηλώσεων (Schema).
Β) Απλοί και σύνθετοι τύποι δεδομένων (Simple and Complex Type Definitions).
Γ) Δηλώσεις συστατικών (Element Declarations).

Τα δευτερεύοντα συστατικά :

1) Ορισμοί ομάδων ιδιοτήτων (Attribute Group Definitions).
2) Ορισμοί ομοιότητας-περιορισμών (Identity – Constraint Definitions).
3) Ορισμοί ονομασμένων ομάδων (Named Group Definitions).
4) Δηλώσεις σχολίων (Notation Declarations).

Τα βοηθητικά συστατικά, τα οποία χρησιμεύουν στα υπόλοιπα συστατικά και δεν μπορούν να υπάρξουν μόνα τους:

- Ομάδες αντικατάστασης.
- Σχόλια.

Πτυχιακή εργασία της φοιτήτριας Αικατερίνης-Ιωσηφίνα Αρβανιτάκη

– Χαρακτήρες.

Τύποι δεδομένων του σχήματος XML (Schema Datatypes)

Πρόκειται για το δεύτερο μέρος του ορισμού του σχήματος XML. Προτείνει λειτουργίες για ορισμό τύπων δεδομένων προς χρήση, για περιορισμό των τύπων δεδομένων των στοιχείων και ιδιοτήτων, μέσα στα σχήματα XML.

Παρέχει:

- Ένα σύνολο ενσωματωμένων πρωταρχικών τύπων δεδομένων.

- Ένα σύνολο ενσωματωμένων τύπων δεδομένων που βασίζονται σε άλλους τύπους δεδομένων.

- Μηχανισμούς με τους οποίους οι χρήστες μπορούν να ορίσουν τους δικούς τους τύπους δεδομένων.

Τύποι δεδομένων.

Παροχή ενός συνόλου βασικών τύπων δεδομένων (κείμενο, ακέραιοι και πραγματικοί αριθμοί, ημερομηνίες κλπ), για την επιτυχή περιγραφή σύνθετων δεδομένων όπως ιστογράμματα, γραφικά κ.α καθώς και τύποι δεδομένων πίνακα (Array και Datatypes Matrix), όπου περιλαμβάνονται πίνακες συγκεκριμένου ή παραμετρικού μεγέθους .

Ενσωματωμένοι θεμελιώδεις χρονικοί τύποι δεδομένων (Built-in πρωτογενής τύπους δεδομένων χρόνου).

Περιλαμβάνονται ο τύπος δεδομένων χρονικής στιγμής (basicTimePoint) και η διάρκεια (basicDuration).

Παροχή επαρκούς μοντέλου για σύνδεση ή / και αναφορά ανάμεσα σε έναν ή περισσότερους περιγραφείς και των υπό περιγραφή δεδομένων.

Σε αυτό το κεφάλαιο θα γίνει μια αρχική γνωριμία με το σχήμα XML και τι μπορεί να απαρτίζει ένα σχήμα XML. Θα δούμε ότι αποτελείται από στοιχεία τα οποία και θα δούμε αναλυτικά, καθώς και την σημασία του κάθε στοιχείου XML σχήμα.

# 1.1 ELEMENTS

Ένα[1] στοιχείο XML είναι τα πάντα, από (συμπεριλαμβανομένης) ετικέτα έναρξης του στοιχείου έως (συμπεριλαμβανομένης) ετικέτα τέλους του στοιχείου.

Ένα στοιχείο μπορεί να έχει :

- άλλα στοιχεία
- κείμενο
- γνωρίσματα
- ή ένα μίγμα από όλα τα παραπάνω ...

1) <schema> element

Το στοιχείο <schema> είναι το στοιχείο ρίζα σε κάθε XML Schema
<?xml version="1.0"?>

<xs:schema>
...
...
</xs:schema>

Το <schema> στοιχείο επιτρέπεται να περιέχει κάποια χαρακτηριστικά.
Μια δήλωση σχήματος μοιάζει συχνά κάτι σαν αυτό:

<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
...
...
</xs:schema>

Το παρακάτω
xmlns:xs="http://www.w3.org/2001/XMLSchema"

δηλώνει ότι τα στοιχεία και οι τύποι δεδομένων που χρησιμοποιούνται στο σχήμα προέρχονται από το "http://www.w3.org/2001/XMLSchema" namespace. Επίσης

διευκρινίζεται ότι τα στοιχεία και οι τύποι δεδομένων που προέρχονται από το "http://www.w3.org/2001/XMLSchema" namespace θα πρέπει να έχουν το πρόθεμα xs: .

Αυτή η φράση:
targetNamespace = "http://www.w3schools.com"

δείχνει ότι τα στοιχεία που ορίζονται από αυτό το σχήμα (note, to, from, heading, body.) προέρχεται από το "http://www.w3schools.com" namespace.

Αυτή η φράση:
xmlns="http://www.w3schools.com"

δείχνει ότι το προεπιλεγμένο πεδίο ονομάτων  είναι "http://www.w3schools.com".

Αυτή η φράση:
elementFormDefault = "qualified "

υποδεικνύει ότι τυχόν στοιχεία που χρησιμοποιούνται από το έγγραφο παράδειγμα XML που είχαν δηλωθεί σε αυτό το σχήμα πρέπει να έχουν namespace προσόντα.

Αναφορά ενός σχήματος σε ένα έγγραφο XML

Αυτό το έγγραφο XML έχει μια αναφορά σε ένα σχήμα XML:

```
<?xml version="1.0"?>

<note xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">

<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Το ακόλουθο απόσπασμα:
xmlns = "http://www.w3schools.com"

ορίζεται η προεπιλεγμένη δήλωση ονομάτων.  Αυτή η δήλωση λέει στο σχήμα-validator ότι όλα τα στοιχεία που χρησιμοποιούνται σε αυτό το έγγραφο XML που δηλώθηκαν στην "http://www.w3schools.com" namespace.

Μόλις έχετε το XML Schema Instance  namespace διαθέσιμο:
xmlns: XSI = "http://www.w3.org/2001/XMLSchema-instance"

μπορείτε να χρησιμοποιήσετε το χαρακτηριστικό schemaLocation. Αυτό το χαρακτηριστικό

έχει δύο τιμές. Η πρώτη τιμή είναι το namespace που χρησιμοποιείται. Η δεύτερη αξία είναι η θέση του σχήματος XML που θα χρησιμοποιήσετε για εκείνη την περιοχή:
XSI: schemaLocation = "http://www.w3schools.com note.xsd"

## 1.2 XSD Simple Elements

Τα XML Schemes ορίζουν τα στοιχεία του XML αρχείου σας.
Ένα απλό στοιχείο είναι ένα στοιχείο XML που περιέχει μόνο κείμενο. Δεν μπορεί να περιέχει οποιαδήποτε άλλα στοιχεία ή ιδιότητες.

Τι είναι ένα απλό στοιχείο;

Ένα απλό στοιχείο είναι ένα στοιχείο XML που μπορεί να περιέχει μόνο κείμενο. Δεν μπορεί να περιέχει οποιαδήποτε άλλα στοιχεία ή ιδιότητες.

Ωστόσο, ο "μόνο κείμενο" περιορισμός είναι αρκετά παραπλανητικός. Το κείμενο μπορεί να είναι πολλών διαφορετικών τύπων. Μπορεί να είναι ένα από τα είδη που περιλαμβάνονται στο XML Schema ορισμός (boolean, string, ημερομηνία, κ.λπ.), ή μπορεί να είναι μια προσαρμοσμένου τύπου που μπορείτε να ορίσετε μόνοι σας.

Μπορείτε επίσης να προσθέσετε περιορισμούς (έδρες) σε έναν τύπο δεδομένων, προκειμένου να περιορίσει το περιεχόμενό της, ή μπορείτε να απαιτείτε τα δεδομένα να ταιριάζουν με ένα συγκεκριμένο μοτίβο.

Καθορισμός ενός απλού στοιχείου

Η σύνταξη για τον καθορισμό ενός απλού στοιχείου είναι:
<xs:element name="xxx" type="yyy"/>

- xs:decimal(τύπος στοιχείου δεκαδικός αριθμός)
- xs:integer (τύπος στοιχείου ακέραιος αριθμός)
- xs:boolean (τύπος στοιχείου  boolean δηλαδή μπορεί να πάρει τις τιμές true ή false)
- xs:date (τύπος στοιχείου ημερομηνία)
- xs:time (τύπος στοιχείου ώρα)
- xs:string (τύπος στοιχείου χαρακτήρες)

Παραδείγματα :

XML elements:

<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>

Και εδώ είναι οι αντίστοιχοι απλοί ορισμοί στοιχείου:

<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>

Μια default τιμή αποδίδεται αυτόματα στο αντίστοιχο στοιχείο
όταν καμία άλλη τιμή δεν έχει καθοριστεί.

<xs:element name="color" type="xs:string" default="red"/>

Μια fixed τιμή αποδίδεται αυτόματα στο αντίστοιχο στοιχείο
όταν καμία άλλη τιμή δεν έχει καθοριστεί.

<xs:element name="color" type="xs:string" fixed="red"/>

## 1.3 XSD Attributes

Όλα τα χαρακτηριστικά δηλώνονται ως απλοί τύποι.
Απλά στοιχεία δεν μπορούν να έχουν ιδιότητες. Αν ένα στοιχείο έχει ιδιότητες, θεωρείται ότι είναι σύνθετος τύπος. Αλλά το χαρακτηριστικό το ίδιο δηλώνεται πάντα ως απλός τύπος.

Η σύνταξη για τον καθορισμό ενός χαρακτηριστικού είναι:

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

- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Παράδειγμα:

<lastname lang="EN">Smith</lastname>

Και εδώ είναι ο αντίστοιχος ορισμός χαρακτηριστικού:
<xs:attribute name="lang" type="xs:string"/>

Περιορισμοί σχετικά με τις αξίες

Παραδείγματα:

α) 0<age<120
<xs:element name="age">
<xs:simpleType>
<xs:restriction base="xs:integer">

```
<xs:minInclusive value="0"/>
<xs:maxInclusive value="120"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

β) ορίζει ένα στοιχείο που ονομάζεται «car» με έναν περιορισμό.
Οι μόνες αποδεκτές τιμές είναι: Audi, Golf, BMW:

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

| ***Περιορισμοί*** | ***Επεξήγηση*** |
|---|---|
| enumeration | Ορίζει μια λίστα των αποδεκτών τιμών . |
| fractionDigits | Καθορίζει το μέγιστο αριθμό των επιτρεπόμενων δεκαδικών ψηφίων. Πρέπει να είναι ίση ή μεγαλύτερη από το μηδέν . |
| length | Καθορίζει τον ακριβή αριθμό των επιτρεπόμενων χαρακτήρων ή των στοιχείων λίστας. Πρέπει να είναι ίση ή μεγαλύτερη από το μηδέν. |
| maxExclusive / minExclusive | Καθορίζει τα άνω /κάτω φράγματα για αριθμητικές τιμές (η τιμή πρέπει να είναι μικρότερη από την τιμή αυτή) |
| maxInclusive / minInclusive | Καθορίζει τα άνω /κάτω φράγματα για αριθμητικές τιμές (η τιμή πρέπει να είναι μικρότερη ή ίση με την τιμή αυτή) |
| maxLength/ minLength | Καθορίζει το μέγιστο/ελάχιστο αριθμό των επιτρεπόμενων χαρακτήρων ή των στοιχείων λίστας. Πρέπει να είναι ίση ή μεγαλύτερη από το μηδέν. |
| pattern | πατέντα |
| totalDigits | Συνολικός αριθμός χαρακτήρων. |

Στο παραπάνω πίνακα υπάρχουν όλοι οι δυνατοί περιορισμοί που μπορούμε να θέσουμε σε κάποιο στοιχείο και οι επεξηγήσει τους

# 1.4 XSD Complex Elements

Ένα σύνθετο στοιχείο είναι ένα στοιχείο XML που περιέχει άλλα στοιχεία ή / και ιδιότητες.

Υπάρχουν τέσσερα είδη σύνθετων στοιχείων:

I) κενά στοιχεία

II) στοιχεία που περιέχουν μόνο άλλα στοιχεία
III) στοιχεία που περιέχουν μόνο κείμενο
IV) στοιχεία που περιέχουν και άλλα στοιχεία και το κείμενο

Σημείωση: Κάθε ένα από αυτά τα στοιχεία μπορεί να περιέχουν και χαρακτηριστικά!
Παραδείγματα σύνθετων στοιχείων

Ένα σύνθετο στοιχείο XML, "product ", το οποίο είναι άδειο:
<product pid="1345"/>

Ένα σύνθετο στοιχείο XML, «employee», το οποίο περιέχει μόνο άλλα στοιχεία:
```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

Μπορούμε να ορίσουμε ένα σύνθετο στοιχείο σε ένα σχήμα XML με δύο διαφορετικούς τρόπους:

1. Το «employee» στοιχείο μπορεί να δηλωθεί άμεσα από την ονομασία του στοιχείου, όπως αυτό:

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

2. Το "employee" στοιχείο μπορεί να έχει ένα χαρακτηριστικό τύπο(type), ο οποίος τύπος είναι τύπου complexType:

```
<xs:element name="employee" type="personinfo"/>

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

Εάν χρησιμοποιείτε τη μέθοδο που περιγράφεται παραπάνω, ορισμένα στοιχεία μπορεί να αναφέρονται στον ίδιο σύνθετο τύπο, όπως τα παρακάτω:

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
```

```
<xs:element name="member" type="personinfo"/>

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

Μπορείτε να βασίσετε επίσης ένα σύνθετο στοιχείο σε ένα υπάρχον σύνθετο στοιχείο και να προσθέσετε κάποια στοιχεία, όπως αυτό:

```
<xs:element name="employee" type="fullpersoninfo"/>

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

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

## 1.5 Complex Empty Elements

Ένα άδειο complex στοιχείο δεν μπορεί να έχει περιεχόμενο, μόνο χαρακτηριστικά.

Ένα άδειο”XML element:

```
<product prodid="1345" />
```

Το «product» στοιχείο παραπάνω δεν έχει περιεχόμενο σε όλα. Για να ορίσετε έναν τύπο χωρίς περιεχόμενο, πρέπει να ορίσουμε ένα είδος που επιτρέπει στα στοιχεία ως προς το περιεχόμενο, αλλά δεν δηλώνουν στην πραγματικότητα οποιαδήποτε στοιχεία, όπως αυτό:

```
<xs:element name="product">
  <xs:complexType>
    <xs:complexContent>
      <xs:restriction base="xs:integer">
       <xs:attribute name="prodid" type="xs:positiveInteger"/>
      </xs:restriction>
```

```
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

Στο παραπάνω παράδειγμα, ορίζουμε ένα σύνθετο τύπο με ένα σύνθετο περιεχόμενο. Το στοιχείο complexContent δείχνει την πρόθεση να περιορίσει ή να επεκτείνει το περιεχόμενο ενός σύνθετου τύπου, και ο περιορισμός του ακεραίου δηλώνει ένα χαρακτηριστικό, αλλά δεν θεσπίζει κάποιο στοιχείο.

Ωστόσο, είναι δυνατόν να δηλώσει το «προϊόν» στοιχείο πιο συμπαγώς, όπως αυτό:

```
<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

Ή μπορείτε να δώσετε στο complexType στοιχείο ένα όνομα, όπως παρακάτω το στοιχείο "προϊόν" έχει ένα χαρακτηριστικό τύπο, complexType (αν χρησιμοποιείτε αυτή τη μέθοδο, αρκετά στοιχεία μπορούν να αναφέρονται στον ίδιο τύπο):

```
<xs:element name="product" type="prodtype"/>

<xs:complexType name="prodtype">
  <xs:attribute name="prodid" type="xs:positiveInteger"/>
</xs:complexType>
```

## 1.6 Complex Types Containing Elements Only

Σύνθετους τύπους στοιχείων που περιλαμβάνουν μόνο στοιχεία.

Ένα στοιχείο XML, "person ", που περιέχει μόνο άλλα στοιχεία:

```
<person>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</person>
```

Μπορείτε να ορίσετε το «person» στοιχείο σε ένα σχήμα, όπως αυτό:

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

ετικέτα <xs:sequence>. Αυτό σημαίνει ότι τα στοιχεία που ορίζονται ("firstname" και "lastname") πρέπει να εμφανίζονται με αυτή τη σειρά μέσα σε ένα «person » στοιχείο.

```
<xs:element name="person" type="persontype"/>
```

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

## 1.7 Complex Text-Only Elements

Αυτός ο τύπος περιέχει μόνο απλό περιεχόμενο (κείμενο και ιδιότητες), γι 'αυτό προσθέστε ένα στοιχείο simpleContent γύρω από το περιεχόμενο. Όταν χρησιμοποιείτε απλό περιεχόμενο, πρέπει να ορίσετε μια επέκταση ή περιορισμό μέσα στο στοιχείο simpleContent, όπως αυτό:

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
  <xs:extension base="basetype">
     ...
    </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

OR

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="basetype">
       ....
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

## 1.8 Complex Types with Mixed Content

Ένα στοιχείο XML, «letter », το οποίο περιέχει κείμενο και άλλα στοιχεία:

```
<letter>
 Dear Mr.<name>John Smith</name>.
 Your order <orderid>1032</orderid>
 will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

"letter" element:

```
<xs:element name="letter">
 <xs:complexType mixed="true">
```

```
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Σημείωση: Για να ενεργοποιήσετε τα στοιχεία του χαρακτήρα που θα εμφανίζεται μεταξύ του παιδιού-στοιχεία της «letter », το μικτό χαρακτηριστικό πρέπει να ρυθμιστεί σε "true". Η ετικέτα <xs:sequence> σημαίνει ότι τα στοιχεία που ορίζονται (name , OrderID και shipdate) πρέπει να εμφανίζονται με αυτή τη σειρά μέσα σε ένα «letter » στοιχείο.

Θα μπορούσαμε επίσης να δώσουμε στο complexType στοιχείο ένα όνομα, παρακάτω το στοιχείο "letter " έχει ένα χαρακτηριστικό τύπο που αναφέρεται στο όνομα του complexType (αν χρησιμοποιείτε αυτή τη μέθοδο, αρκετά στοιχεία μπορούν να αναφέρονται στο ίδιο συγκρότημα τύπου):

```
<xs:element name="letter" type="lettertype"/>

<xs:complexType name="lettertype" mixed="true">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="orderid" type="xs:positiveInteger"/>
  <xs:element name="shipdate" type="xs:date"/>
 </xs:sequence>
</xs:complexType>
```

## 1.9 XSD Indicators

Μπορούμε να ελέγξουμε το ΠΩΣ τα στοιχεία πρέπει να χρησιμοποιούνται σε έγγραφα με δείκτες.

Υπάρχουν επτά είδη δεικτών τα οποία χωρίζονται σε 3 κατηγορίες:

α)Order indicators:

- All
- Choice
- Sequence

β)Occurrence indicators:

- maxOccurs
- minOccurs

γ)Group indicators:

- Group name
- attributeGroup name

Order Indicators

### All Indicator

Ο <all> δείκτης προσδιορίζει ότι τα στοιχεία που το παιδί μπορούν να εμφανιστούν σε οποιαδήποτε σειρά, και ότι κάθε στοιχείο-παιδί πρέπει να συμβεί μόνο μία φορά:

```
<xs:element name="person">
  <xs:complexType>
   <xs:all>
     <xs:element name="firstname" type="xs:string"/>
     <xs:element name="lastname" type="xs:string"/>
   </xs:all>
  </xs:complexType>
</xs:element>
```

### Choice Indicator

Ο δείκτης <choice> ορίζει ότι είτε ένα στοιχείο ή ένα άλλο παιδί μπορεί να συμβεί:

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
       <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

### Sequence Indicator

Ο δείκτης <sequence> διευκρινίζει ότι τα στοιχεία-παιδιά πρέπει να εμφανίζονται σε μια συγκεκριμένη σειρά:

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

# Occurrence Indicators

### maxOccurs Indicator

Ο δείκτης <maxOccurs> καθορίζει τον μέγιστο αριθμό των φορών που ένα στοιχείο μπορεί να συμβεί:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
```

```
    <xs:element name="full_name" type="xs:string"/>
    <xs:element name="child_name" type="xs:string" maxOccurs="10"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>
```

## minOccurs Indicator

Ο δείκτης <minOccurs> καθορίζει τον ελάχιστο αριθμό των φορών που ένα στοιχείο μπορεί να συμβεί:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
      maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

“Για να επιτρέψετε σε ένα στοιχείο να εμφανιστεί για απεριόριστο αριθμό φορών, χρησιμοποιήστε το maxOccurs = "απεριόριστη" δήλωση: "

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<persons xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="family.xsd">

<person>
  <full_name>Hege Refsnes</full_name>
  <child_name>Cecilie</child_name>
</person>

<person>
  <full_name>Tove Refsnes</full_name>
  <child_name>Hege</child_name>
  <child_name>Stale</child_name>
  <child_name>Jim</child_name>
  <child_name>Borge</child_name>
</person>

<person>
  <full_name>Stale Refsnes</full_name>
</person>

</persons>
```

Το αρχείο XML από πάνω περιέχει ένα στοιχείο ρίζα που ονομάζεται "persons ". Μέσα σε αυτό το στοιχείο ρίζας έχουμε ορίσει τρεις «persons » στοιχεία. Κάθε «persons » στοιχείο πρέπει να περιλαμβάνει "full_name» στοιχείο και μπορεί να περιέχει έως και πέντε

"child_name» στοιχεία.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

<xs:element name="persons">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="person" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="full_name" type="xs:string"/>
            <xs:element name="child_name" type="xs:string"
          minOccurs="0" maxOccurs="5"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

# Group Indicators

## Element Groups

```
<xs:group name="groupname">
...
</xs:group>
```

Πρέπει να ορίσετε μια all, επιλογή, ή ακολουθία Element στο εσωτερικό της δήλωσης της ομάδας. Το ακόλουθο παράδειγμα ορίζει μια ομάδα με το όνομα "persongroup", που καθορίζει μια ομάδα στοιχείων που πρέπει να συμβεί σε μια ακριβή ακολουθία:

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>
```

Αφού ορίσετε μια ομάδα, μπορείτε να την κάνετε αναφορά σε ένα άλλο ορισμό, όπως αυτό:

```
<xs:group name="persongroup">
```

```
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="person" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

### Attribute Groups

```
<xs:attributeGroup name="groupname">
...
</xs:attributeGroup>
```

group named "personattrgroup":

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>
```

Ένα attribute group, **μπορείτε να την κάνετε αναφορά σε ένα άλλο ορισμό, όπως αυτό:**

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>

<xs:element name="person">
  <xs:complexType>
    <xs:attributeGroup ref="personattrgroup"/>
  </xs:complexType>
</xs:element>
```

## 1.10 The <any> Element

Το στοιχείο <Any> μας δίνει τη δυνατότητα να επεκταθεί το XML έγγραφο με στοιχεία που δεν προσδιορίζονται από το σχήμα.
Το παράδειγμα που ακολουθεί είναι ένα απόσπασμα από ένα σχήμα XML που ονομάζεται

"family.xsd". Δείχνει μια δήλωση για το «person » στοιχείο. Με τη χρήση του <Any> στοιχείο που μπορούμε να επεκτείνουμε (μετά το <lastname>) το περιεχόμενο του «person » με οποιοδήποτε στοιχείο:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Τώρα θέλουμε να επεκτείνουμε το «person » στοιχείο με το «children » στοιχείο. Σε αυτή την περίπτωση μπορούμε να το κάνουμε, ακόμα κι αν ποτέ ο συγγραφέας του παραπάνω σχήματος δεν δηλώσε «children » στοιχεία.

Κοιτάξτε αυτό το αρχείο σχήματος, που ονομάζεται "children.xsd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="children">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="childname" type="xs:string"
      maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Το αρχείο XML που ακολουθεί (που ονομάζεται "Myfamily.xml»), χρησιμοποιεί στοιχεία από δύο διαφορετικά σχήματα? "Family.xsd" και "children.xsd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<persons xmlns="http://www.microsoft.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.microsoft.com family.xsd
http://www.w3schools.com children.xsd">

<person>
  <firstname>Hege</firstname>
```

```
  <lastname>Refsnes</lastname>
  <children>
    <childname>Cecilie</childname>
  </children>
</person>

<person>
  <firstname>Stale</firstname>
  <lastname>Refsnes</lastname>
</person>

</persons>
```

Το αρχείο XML από πάνω είναι “valid”, διότι το σχήμα "family.xsd" μας επιτρέπει να επεκταθεί το «person » στοιχείο με ένα προαιρετικό στοιχείο, μετά το «lastname » στοιχείο.

Τα <Any> και <anyAttribute> στοιχεία χρησιμοποιούνται για να κάνουν Extensible έγγραφα! Επιτρέπουν στα έγγραφα να περιέχουν πρόσθετα στοιχεία που δεν έχουν δηλωθεί στο κύριο σχήμα XML.

## 1.11 The <anyAttribute> Element

Το στοιχείο <anyAttribute> μας δίνει τη δυνατότητα να επεκταθεί το έγγραφο XML με ιδιότητες που δεν καθορίζει το σχήμα.
Το παράδειγμα που ακολουθεί είναι ένα απόσπασμα από ένα σχήμα XML που ονομάζεται "family.xsd". Δείχνει μια δήλωση για το «person» στοιχείο. Με τη χρήση του στοιχείου <anyAttribute> μπορούμε να προσθέσουμε οποιοδήποτε αριθμό χαρακτηριστικών για το «person» στοιχείο:

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

Τώρα θέλουμε να επεκτείνουμε το «πρόσωπο» στοιχείο με "φύλο" χαρακτηριστικό. Σε αυτή την περίπτωση μπορούμε να το κάνουμε, ακόμα κι αν ποτέ ο συγγραφέας του παραπάνω σχήματος που δηλώνονται κάθε "φύλο" χαρακτηριστικό.

Κοιτάξτε αυτό το αρχείο σχήματος, που ονομάζεται "attribute.xsd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
```

```
elementFormDefault="qualified">

<xs:attribute name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
    <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

</xs:schema>
```

Το αρχείο XML που ακολουθεί (που ονομάζεται "Myfamily.xml»), χρησιμοποιεί στοιχεία από δύο διαφορετικά σχήματα? "Family.xsd" και "attribute.xsd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<persons xmlns="http://www.microsoft.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://www.microsoft.com family.xsd
http://www.w3schools.com attribute.xsd">

<person gender="female">
  <firstname>Hege</firstname>
  <lastname>Refsnes</lastname>
</person>

<person gender="male">
  <firstname>Stale</firstname>
  <lastname>Refsnes</lastname>
</person>

</persons>
```

Το παραπάνω αρχείο XML ισχύει, διότι το σχήμα "family.xsd" επιτρέπει να προσθέσετε ένα χαρακτηριστικό για το «person» στοιχείο.

Η <Any> και στοιχεία <anyAttribute> χρησιμοποιούνται για να κάνουν Extensible έγγραφα! Επιτρέπουν έγγραφα που να περιέχουν πρόσθετα στοιχεία που δεν έχουν δηλωθεί στην κύρια σχήμα XML.

## 1.12 Element Substitution

Ας πούμε ότι έχουμε χρήστες από δύο διαφορετικές χώρες: Αγγλία και Νορβηγία. Θα θέλαμε τη δυνατότητα να επιτρέπουν στο χρήστη να επιλέξει εάν αυτός ή αυτή θα ήθελε να χρησιμοποιήσει τα νορβηγικά ονόματα των στοιχείων ή τα αγγλικά ονόματα στοιχείο στο έγγραφο XML.

Για να λυθεί αυτό το πρόβλημα, θα μπορούσαμε να καθορίσει μια substitutionGroup στο

σχήμα XML. Κατ 'αρχάς, δηλώνουμε ένα στοιχείο το root και στη συνέχεια δηλώνουμε τα άλλα στοιχεία τα οποία δηλώνουν ότι μπορούν να υποκατασταθούν για το στοιχείο κεφάλι.

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

Στο παραπάνω παράδειγμα, το "name " στοιχείο είναι το στοιχείο root και το "navn " στοιχείο μπορεί να υποκατασταθεί με "name ".

Κοιτάξτε αυτό το σχήμα XML:

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

<xs:complexType name="custinfo">
  <xs:sequence>
    <xs:element ref="name"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="customer" type="custinfo"/>
<xs:element name="kunde" substitutionGroup="customer"/>
```

Ένα valid XML document (σύμφωνα με το schema παραπάνω):

```
<customer>
  <name>John Smith</name>
</customer>
```

ή

```
<kunde>
  <navn>John Smith</navn>
</kunde>
```

**Blocking Element Substitution**

Για να αποφύγετε άλλα στοιχεία να αντικατασταθούν με ένα συγκεκριμένο στοιχείο, χρησιμοποιήστε το χαρακτηριστικό μπλοκ:

```
<xs:element name="name" type="xs:string" block="substitution"/>

<xs:element name="name" type="xs:string" block="substitution"/>
<xs:element name="navn" substitutionGroup="name"/>

<xs:complexType name="custinfo">
  <xs:sequence>
    <xs:element ref="name"/>
  </xs:sequence>
</xs:complexType>
```

```
<xs:element name="customer" type="custinfo" block="substitution"/>
<xs:element name="kunde" substitutionGroup="customer"/>
```

Ένα valid XML document (σύμφωνα με το schema παραπάνω):

```
<customer>
  <name>John Smith</name>
</customer>
```

ΑΛΛΑ ΑΥΤΟ ΔΕΝ ΕΊΝΑΙ ΠΛΕΟΝ VALID:

```
<kunde>
  <navn>John Smith</navn>
</kunde>
```

ΠΑΡΑΔΕΙΓΜΑΤΑ

"shiporder.xml":

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

Το έγγραφο XML παραπάνω αποτελείται από ένα στοιχείο ρίζας, "shiporder", που περιέχει ένα απαιτούμενο χαρακτηριστικό που ονομάζεται "orderid ". Το "shiporder» στοιχείο περιέχει τρία διαφορετικά στοιχεία παιδιά: «orderperson", "shipto" και "item ". Το "item " στοιχείο εμφανίζεται δύο φορές, και περιέχει ένα "title ", ένα προαιρετικό «note», ένα «quantity », και ένα «price » στοιχείο.
Η παραπάνω γραμμή: xmlns: XSI = "http://www.w3.org/2001/XMLSchema-instance" λέει στο XML parser ότι το έγγραφο αυτό θα πρέπει να επικυρωθεί σε σχέση με ένα σχήμα. Η

γραμμή: XSI: noNamespaceSchemaLocation = "shiporder.xsd" Ορίζει σε ποιες περιπτώσεις το σχήμα κατοικεί (εδώ είναι στο ίδιο φάκελο με το "shiporder.xml").

"shiporder.xsd"):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- definition of simple elements -->
<xs:element name="orderperson" type="xs:string"/>
<xs:element name="name" type="xs:string"/>
<xs:element name="address" type="xs:string"/>
<xs:element name="city" type="xs:string"/>
<xs:element name="country" type="xs:string"/>
<xs:element name="title" type="xs:string"/>
<xs:element name="note" type="xs:string"/>
<xs:element name="quantity" type="xs:positiveInteger"/>
<xs:element name="price" type="xs:decimal"/>

<!-- definition of attributes -->
<xs:attribute name="orderid" type="xs:string"/>

<!-- definition of complex elements -->
<xs:element name="shipto">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="address"/>
      <xs:element ref="city"/>
      <xs:element ref="country"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="item">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="title"/>
      <xs:element ref="note" minOccurs="0"/>
      <xs:element ref="quantity"/>
      <xs:element ref="price"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
```

```
      <xs:element ref="orderperson"/>
      <xs:element ref="shipto"/>
      <xs:element ref="item" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute ref="orderid" use="required"/>
  </xs:complexType>
</xs:element>

</xs:schema>
```

# ΚΕΦΑΛΑΙΟ 2

# Graphical XML Schemes

# Εισαγωγή

Σε αυτό το κεφάλαιο δείχνω τα πιο σημαντικα graphical XML Schemes και γίνεται η σύγκριση τους. Καθένα από αυτά είναι ξεχωριστό από τα άλλα, καθώς υπάρχουν πολλοί τρόποι αναπαράστασης τους.
Τα προβλήματα που αντιμετωπίζουμε όταν εργαζόμαστε με XSD μπορεί να μετριαστεί με τη χρήση γραφικών εργαλείων επεξεργασίας. Παρά το γεγονός ότι κάθε text-based επεξεργαστής μπορεί να χρησιμοποιηθεί για την επεξεργασία ενός XML Schema, ένα γραφικό editor προσφέρει πλεονεκτήματα, καθώς μας δίνεται η δυνατότητα να "δούμε" την δομή του εγγράφου γραφικά και να επεξεργαστούμε και άλλες χρήσιμες λειτουργίες.

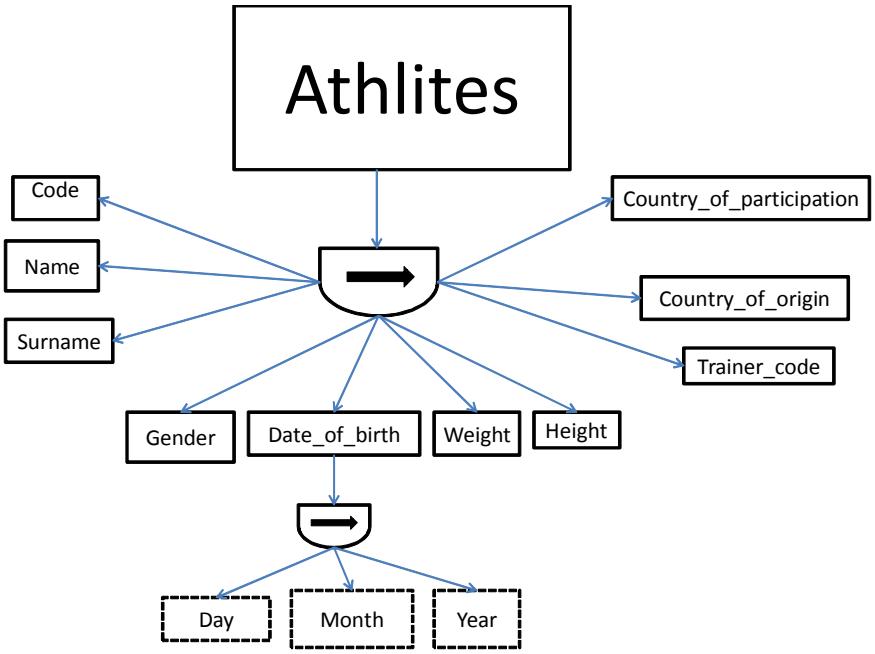
**Γραφική αναπαράσταση XML Schema**[2][3][4]

# 2.1.Cxml

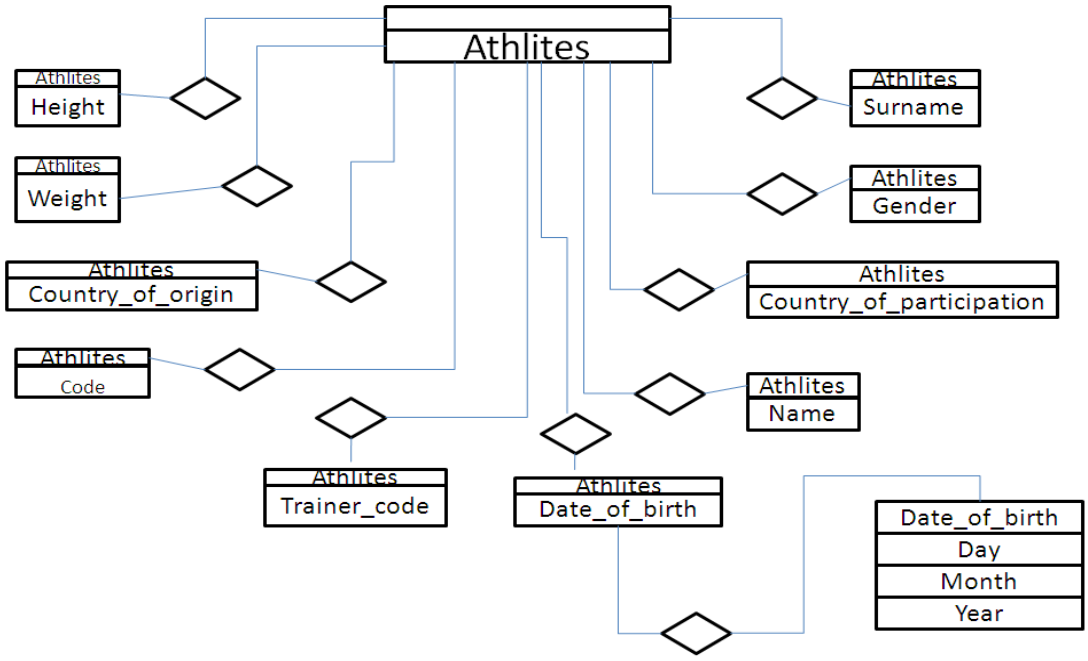
Το Cxml σχήμα εκπροσωπείται σε πλούσιο περιεχόμενο του σχήματος XML που συνήθως λείπει από τα θεωρητικά μοντέλα. Για να κατασκευάσετε ένα Cxml σχήμα ξεκινήστε με το augmented (επαυξημένο) hypergraph του οποίου οι κορυφές και οι ακμές είναι αντικειμένα και σύνολα σχέσεων, αντίστοιχα, που αποτελείται από τα στολίδια που αντιπροσωπεύουν περιορισμούς. Οι γραμμές συνδέουν σύνολα με σύνολα αντικειμένων. Ένας περιορισμός συμμετοχής καθορίζει πόσες φορές ένα αντικείμενο μπορεί να συμμετέχει σε ένα σύνολο σχέσεων.

Περιγραφή της παρακάτω εικόνας:

Κοιτάζοντας το γράφημα1 αρχικά ορίζεται ο κόμβος ρίζα ο οποίος είναι ο "athlites", τότε να εντάξετε  όλα τα πεδία που υπάρχουν μέσα στο "athlites". Βάλτε το βέλος για να δείξετε ότι όλα τα στοιχεία βρίσκονται εντός του "athlites". Τα κουτιά με τις διακεκομμένες γραμμές τα οποία έχουν τοποθετήσει, για να δείξουμε ότι είναι ένα υπο-στοιχείο παράδειγμα "Date_of_birth"



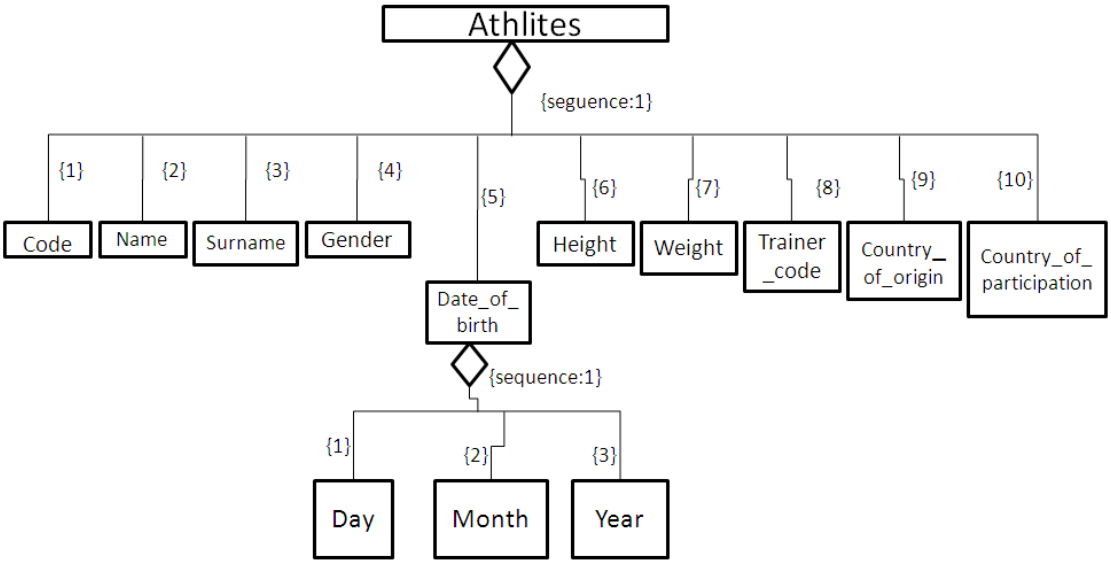

Σχήμα. 1. Δομές Ακολουθίας / Επιλογής (Sequence/Choice Structures)




Σχήμα. 2. Best Representation of Figure 1 using XER Notation.

# 2.2 UML

Χρησιμοποιούμε τα  xml schema graphics XML για να δημιουργήσετε ένα UML.
Στη UML ορίζουν αρχικά το στοιχείο ρίζα του γραφήματος, όπου <xs:sequence> ο τύπος. Στη συνέχεια, τοποθετήστε κάτω από τα υπόλοιπα στοιχεία ενός αθλητή που θα πρέπει να είναι τύπου <xs:sequence>, όπως φαίνεται στην παρακάτω εικόνα. Όταν θέλουμε τα στοιχεία να εμφανίζονται μόνο 1 φορά το καθένα, τότε θα ακολουθία αριθμών: 1. Το διαμάντι δείχνει το στοιχείο ρίζα. Για παράδειγμα, "date_of_birth" στοιχεία-ρίζες είναι: "ημέρα", "μήνας" και "χρόνος".




Εικόνα 3. Καλύτερη Απεικόνιση του Σχήματος 1 Χρησιμοποιώντας Σημειογραφία Κόνραντ

Συγκρίνοντας το C-XML και UML, παρατηρούμε ότι, σύμφωνα με τα κριτήρια για την εννοιολογική μοντελοποίηση XML είναι τα τρία μοντέλα:

- Επίσημη εγκατάστασης: Η c-XML έχει επίσημα σταθερή με σταθερή βάση από την άποψη της λογικής.
- -Αντανάκλαση του νοητικού μοντέλου: εκτός από τα χαρακτηριστικά των οντοτήτων της UML, η XML, ξεχωρίζει. Επίσης, το C-XML αντιπροσωπεύει όλες τις έννοιες ως αντικείμενα-σύνολα κόμβων στα Hypergraphs. Ένα μειονέκτημα είναι ότι ο χρήστης της UML, ή της XML, θα πρέπει να αποφασιστεί από πριν αν θα πρέπει να ορίσει

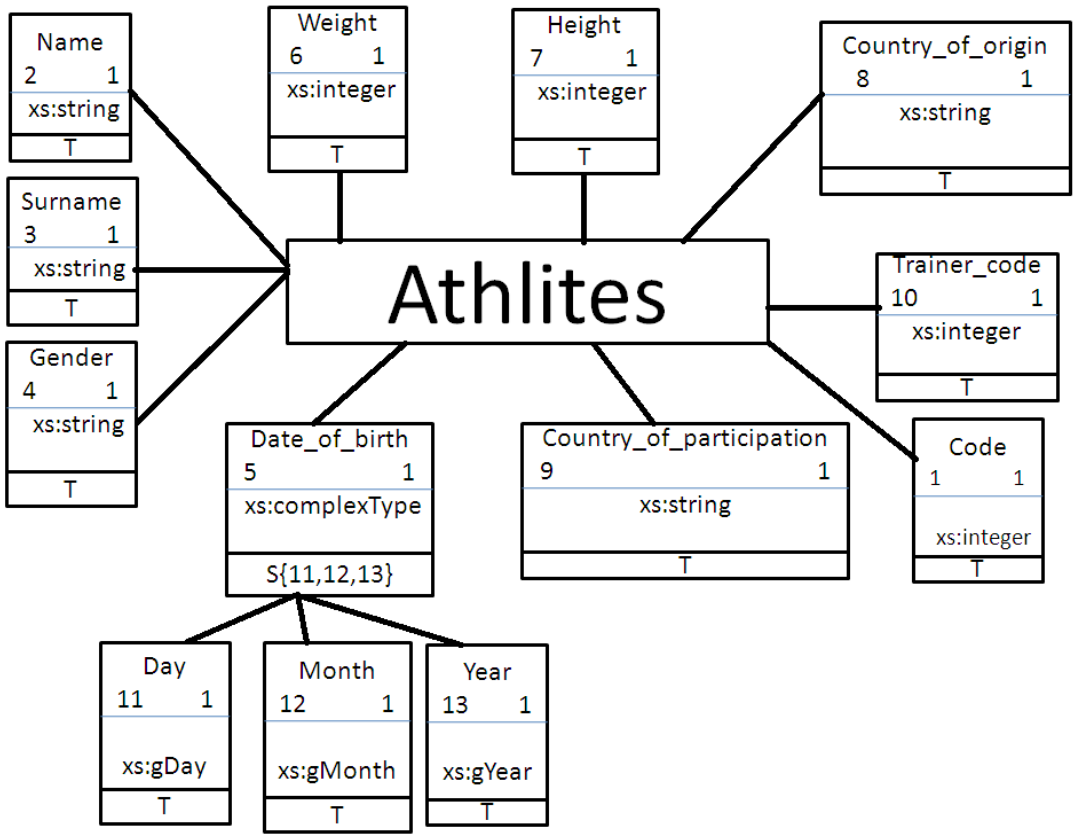
μια οντότητα ή ένα χαρακτηριστικό ως τάξη.

– -Προβολές: Τα Hypergraphs είναι συνήθως πιο δεκτικά σε μεταφράσεις με διάφορους τρόπους, συμπεριλαμβανομένων των εναλλακτικών μορφών XML.

## 2.3 XSD-M

Περιγραφή της εικόνας πιο κάτω:
Στο σχήμα 4 δείτε το "athlites" και όλα τα πεδία. Στο πρώτο (1ο) μέρος βλέπουμε το όνομα του πεδίου, στη δεξιά πλευρά κάτω από το όνομα του πεδίου είναι το νούμερο ένα (1), που δείχνει πόσες φορές μπορεί να είναι το πεδίο που χρησιμοποιείται για κάθε αθλητή. Στην αριστερή πλευρά κάτω από το όνομα του τομέα είναι ο αριθμός που δείχνει την αρίθμηση του πεδίου, ώστε να γνωρίζουν πόσα πεδία υπάρχουν στον αθλητή. Στο δεύτερο (2ο) μέρος του κάθε τομέα γράφουμε τον τύπο. Τέλος, στο τρίτο (3ο) μέρος βάλαμε το T ή S {}. Το T θέλει να μας δείξει ότι κάτω από την περιοχή που δεν ακολουθεί κάτι άλλο, επίσης, ο δείκτης S μας δείχνει ότι μέσα στις αγκύλες {} βάζουμε τους αριθμούς σε πεδία που υπάρχουν μέσα σε αυτό. Για παράδειγμα, "date_of_birth" θέτουμε S {11, 12,13}, όπου 11 είναι η "Ημέρα", το 12 είναι ο "Μήνας" και 13 είναι το "Έτος"

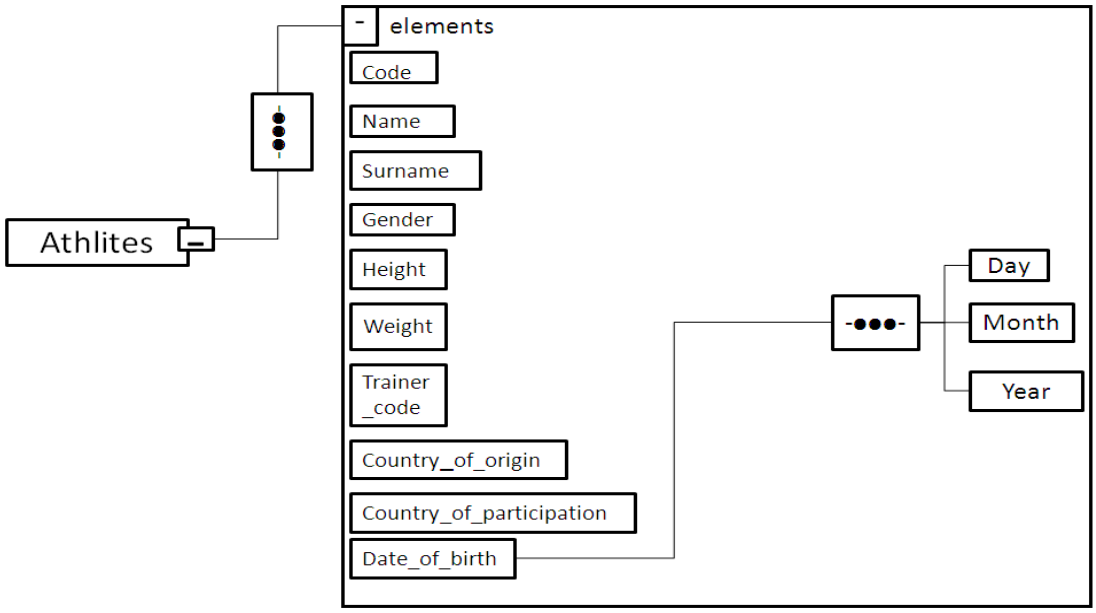
Σχήμα 4. Γραφική αναπαράσταση του δείγματος σε schema XSD-M



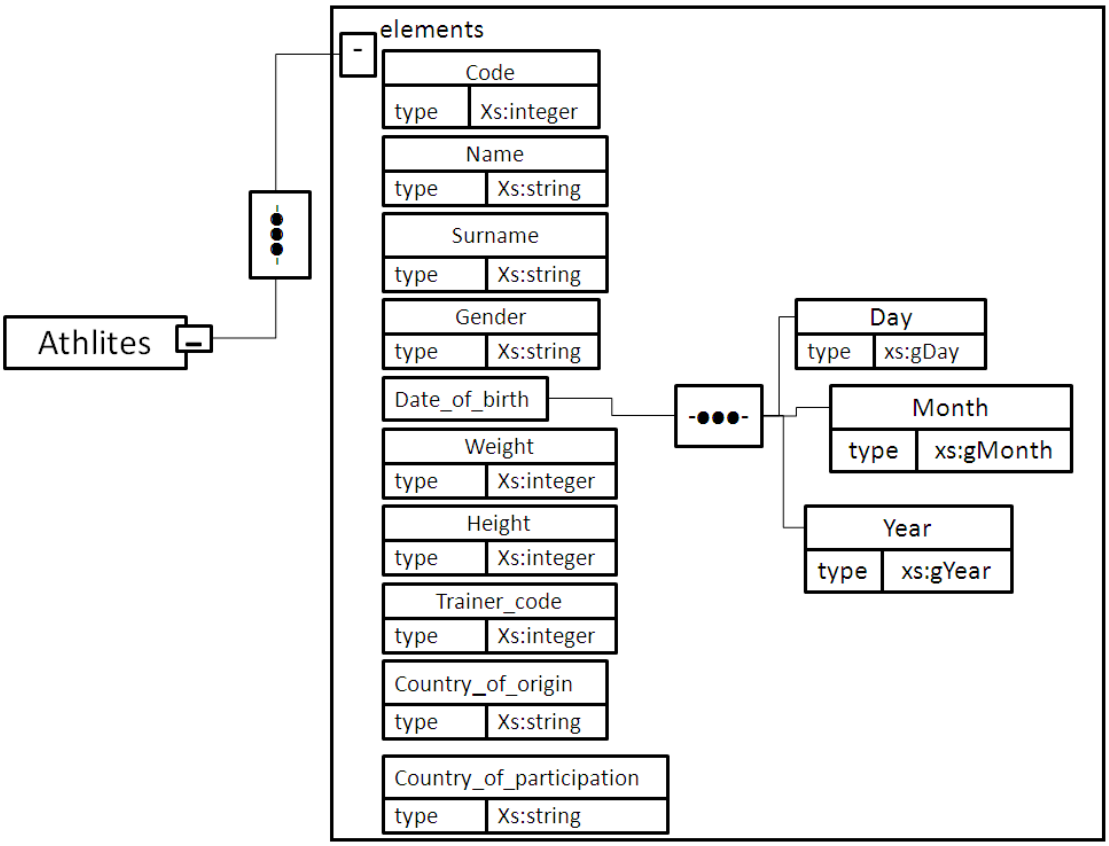
## 2.4 XML Σχημάτων Altova

Για να δημιουργήσετε ένα σχήμα XML altova υπάρχουν δύο (2) εκδόσεις, η πρώτη (1) έκδοση είναι να γράψετε τον κώδικα και να εμφανιστεί το παραπάνω σχέδιο, και η δεύτερη (2) είναι η έκδοση altova να σχεδιάσετε κάθε στοιχείο ξεχωριστά και έτσι δημιουργείται ο κώδικας άμεσα.

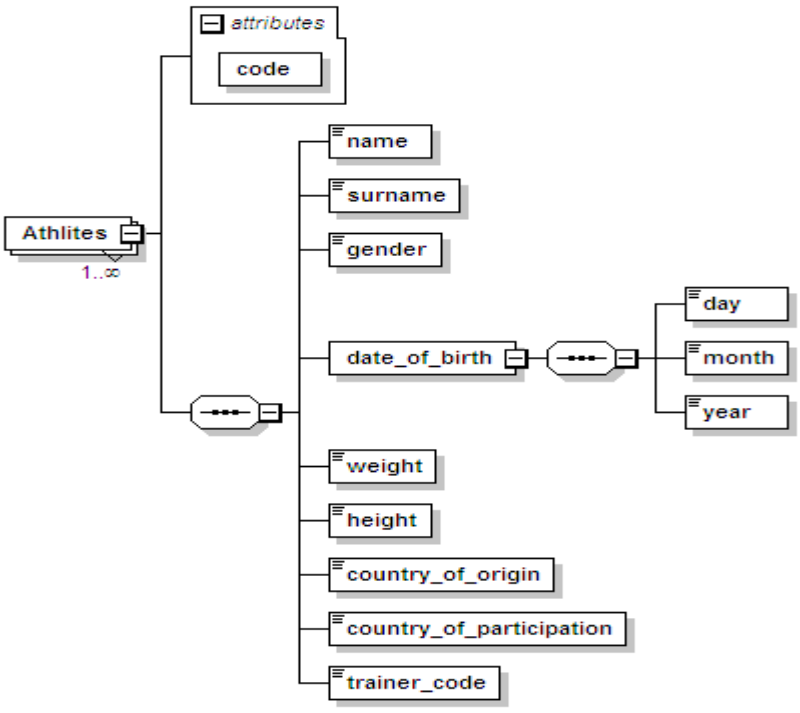
Στο σχήμα 5 βλέπουμε ότι "athlites" είναι ένα σημείο όπου υπάρχουν στοιχεία μέσα σε αυτό. Το σύμβολο-OOO- δηλώνει τον τύπο του complexType.



Επιπλέον, βλέπουμε στην εικόνα 6 οι τύποι του κάθε στοιχείου όπως φαίνεται στο altova. Αυτό μας βοηθά να γνωρίζουμε τον τύπο του κάθε στοιχείου, ώστε να το γράψουμε πιο εύκολα.




Εικόνα 6. Σχήμα altova για το xml “Athlites” με τα στοιχεία του.

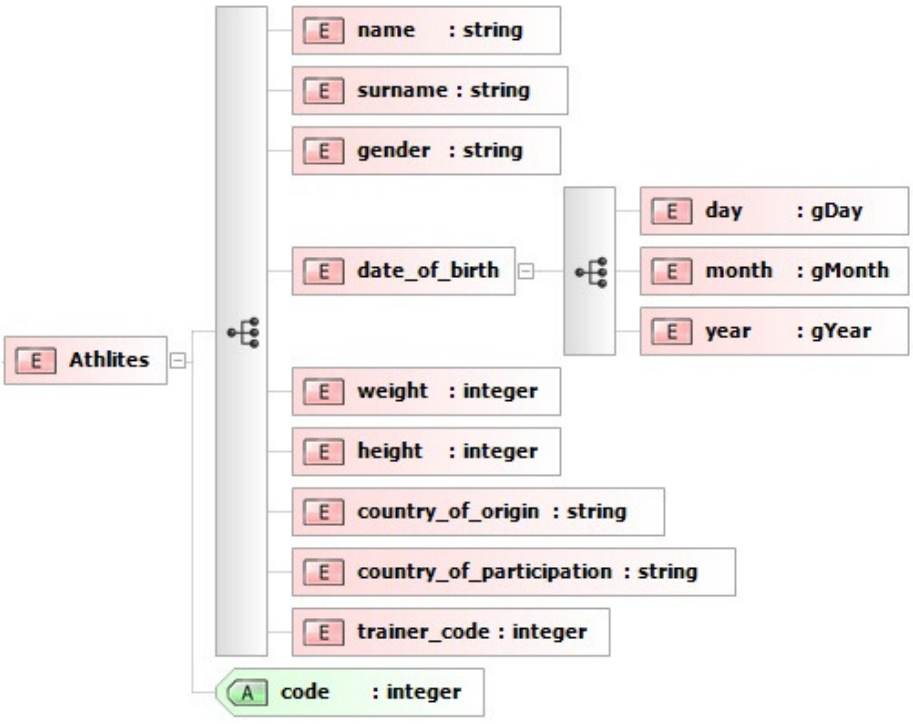


Σχήμα 7. Υπόδειγμα του πεδίου YUMARC πεδίο στοιχείο μορφή και τα χαρακτηριστικά

## 2.5 Liquid

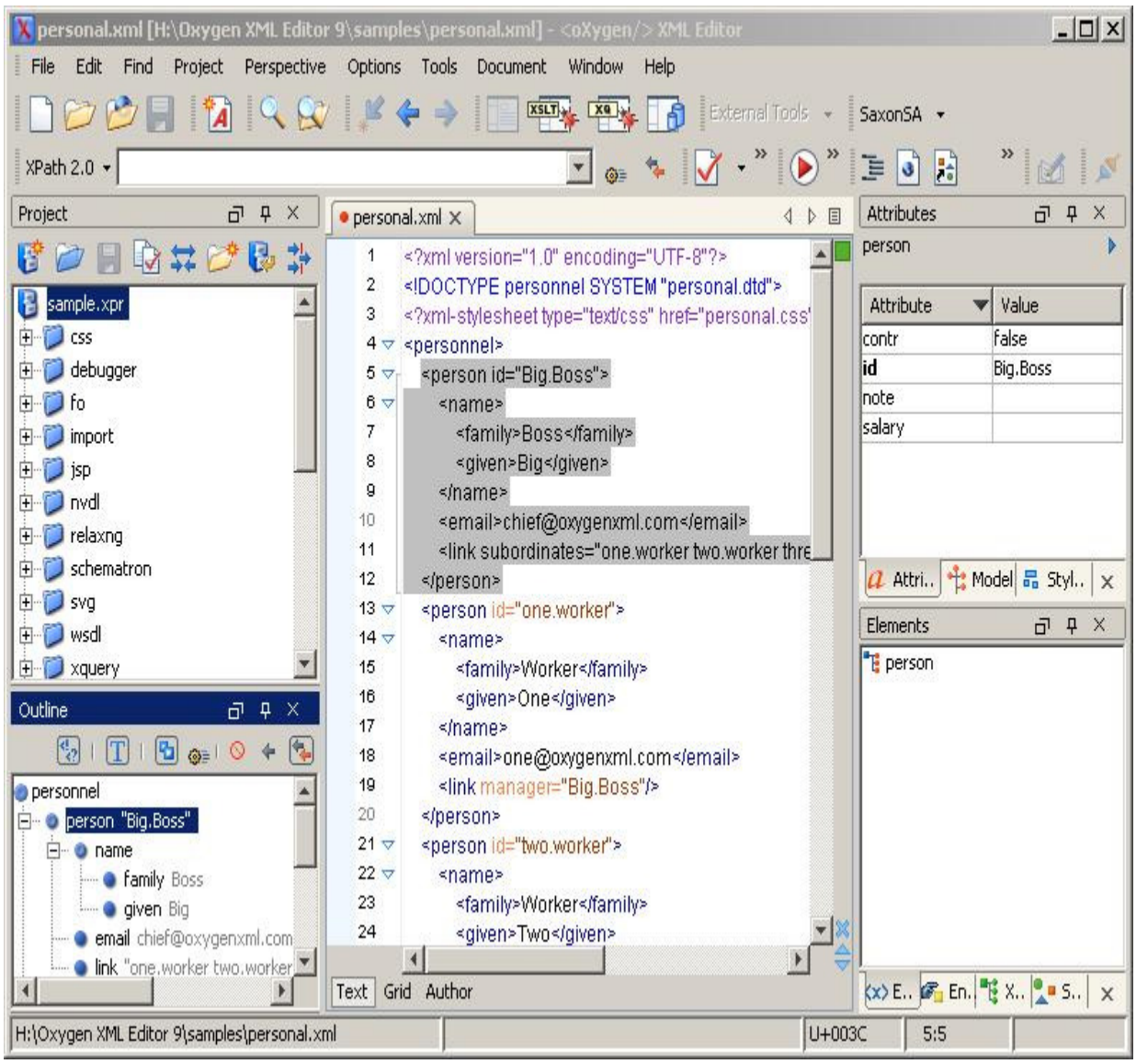
Εξετάζοντας το διάγραμμα πιο κάτω μπορούμε να δούμε ότι υπάρχει «athlites" ως ρίζα και στη συνέχεια ακολουθούν οι λεπτομέρειες του "athlites", με το όνομα, άνω και κάτω τελεία (:) και στη συνέχεια τον τύπο του.

Το σύμβολο αυτό  υποδεικνύει ότι ο τύπος είναι complexType στοιχείο, αποτελείται από τα στοιχεία στα δεξιά του συμβόλου. Επίσης, έχουμε κάποια χαρακτηριστικά  τα οποία εμφανίζονται με ένα «A» μέσα σε ένα πράσινο κουτί.




Σχήμα 8.Liquid

# 2.6 Oxygen XML Editor




Η Oxygen XML Editor είναι μια πολύ-πλατφόρμα επεξεργασίας XML, XSLT / XQuery debugger με υποστήριξη Unicode. Πρόκειται για μια εφαρμογή Java, έτσι ώστε να μπορεί να τρέξει σε Windows, Mac OS X και Linux.

## 2.6.1 XML δυνατότητες επεξεργασίας

Η Oxygen XML προσφέρει μια σειρά από χαρακτηριστικά για την επεξεργασία εγγράφων XML. Γίνεται έλεγχος των εγγράφων για τη σωστή XML μορφή. Μπορούν επίσης να επικυρωθούν από ένα schema. Για σκοπούς επαλήθευσης, τα έγγραφα μπορούν να επικυρωθούν κατά τα schemes DTD, W3C XML Schema, RELAX NG, Schematron, NRL και NVDL. Ο Oxygen XML editor μπορεί να επικυρώσει επίσης τα XML έγγραφα, όπως εισάγονται. Για επιπρόσθετους τύπους σχήματος, ένα σενάριο επικύρωσης μπορεί να παραχθεί, το οποίο επιτρέπει στην Oxygen XML να “καλεί” αυθαίρετα προγράμματα, να κάνουν την επικύρωση.
Επίσης, το πρόγραμμα έχει την υποστήριξη για XML καταλόγους. Ένας κατάλογος XML είναι ένα αρχείο XML, συγκεκριμένης μορφής, που χαρτογραφεί ένα schema definition string σε ένα πραγματικό όνομα του αρχείου στο δίσκο ή στον ιστό(web). Χρησιμοποιώντας καταλόγους επιτρέπεται στο χρήστη να καθορίσει μια διεύθυνση web για ένα schema, αλλά επιτρέπει στην Oxygen να βρεί μια μορφή αρχείου από τη διεύθυνση, εάν ο κατάλογος καθορίζει κάποια μορφή.

Η Oxygen XML προσφέρει τρία είδη εμφανίσεων που έχουν σχεδιαστεί για την επεξεργασία εγγράφων XML. Οι εμφανίσεις αυτές είναι text, grid, and author.

### Εμφάνιση Text

Η εμφάνιση Text είναι η προεπιλεγμένη προβολή για την επεξεργασία ενός εγγράφου XML. Όπως υποδηλώνει το όνομα, η άποψη αυτή δείχνει το XML κείμενο ως κείμενο.

Για τα έγγραφα που σχετίζονται με ένα σχήμα XML, η Oxygen XML προσφέρει ετικέτες ολοκλήρωσης. Η Oxygen XML μπορεί να χρησιμοποιήσει μια σειρά από γλώσσες σχήματος XML, συμπεριλαμβανομένων των DTD, W3C XML Schema, RELAX NG (τόσο συμπαγές και πλήρες).

Εκτός από ετικέτες,οι σχολιασμούς στο schema θα εμφανιστούν ως επεξηγήσεις για τα στοιχεία στα οποία οι σημειώσεις εφαρμόζονται.
Για τα schemes που δεν έχουν ένα πρότυπο μηχανισμό bindingς(binding) με το αρχείο XML, η Oxygen XML παρέχει οδηγίες επεξεργασίας που καθοδηγούν το πρόγραμμα ως προς το ποια schemes να χρησιμοποιήσουν.
Για έγγραφα που δεν έχουν κάποιο schema, η Oxygen μπορεί να αναλύσει τη δομή του εγγράφου και να δημιουργήσουν ένα schema.

### Εμφάνιση Grid

Η Εμφάνιση Grid δείχνει το έγγραφο XML σε ένα τύπου υπολογιστικό φύλλο. Η πιο αριστερή στήλη εμφανίζει τα στοιχεία, συμπεριλαμβανομένων των σχολίων και οδηγίες επεξεργασίας, στο επίπεδο ρίζας. Η επόμενη στήλη δείχνει τις ιδιότητες των στοιχείων ρίζα, και κάθε μοναδικό πρώτο παιδί της ρίζας XML στοιχείου. Εάν το στοιχείο ρίζα έχει έξι παιδιά όλα με το όνομα "section", τότε η προβολή πλέγματος θα δείξει μόνο ένα στοιχείο "section", και μια σημείωση ότι υπάρχουν έξι από αυτούς. Αυτή η επανάληψη συνεχίζει για την επόμενη στήλη.

Αυτή η άποψη δεν είναι συχνά χρήσιμη για HTML ή έγγραφα παρόμοιας μορφής, αλλά μπορεί να είναι χρήσιμη για ορισμένες μορφές XML που μοιάζουν με φύλλα.

Η άποψη αυτή δείχνει την όλη δομή του αρχείου XML. Όλες οι έγγραφες πληροφορίες στο αρχείο θα παρουσιαστούν σε αυτή την άποψη.

**Εμφάνιση Author**

Στην εμφάνιση  Author, XML ετικέτες και χαρακτηριστικά μπορεί να απενεργοποιηθούν πλήρως, ή μπορεί να εμφανιστούν σε διάφορους συνδυασμούς.

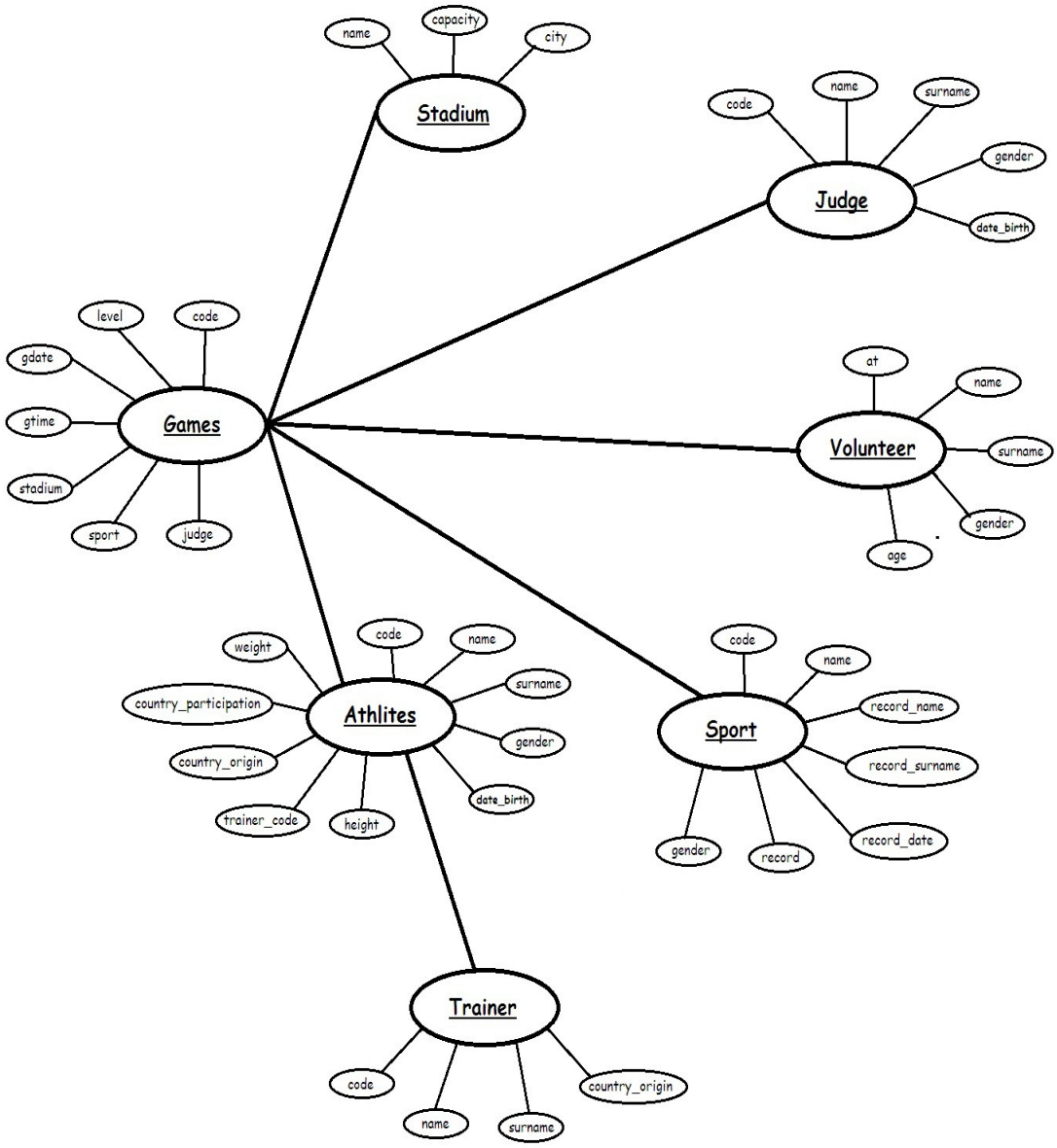
Η επεξεργασία σε αυτή την εμφάνιση είναι ένα ενδιάμεσο βήμα μεταξύ της πραγματικής WYSIWYG επεξεργασίας και της επεξεργασίας στην κανονική text εμφάνιση από την άποψη της πολυπλοκότητας.

Τα στοιχεία XML έχουν γίνει πιο εύκολα στην ανάγνωση από τον άνθρωπο, αλλά οι εμφωλεύσεις και η σημασιολογία του εγγράφου XML είναι ακόμα σαφής.
Ο δρομέας(cursor) μπορεί να τοποθετηθεί μεταξύ οποιωνδήποτε στοιχείων, και όταν η θέση του δρομέα είναι διφορούμενη, ένα εργαλείο-tip παράθυρο θα εμφανιστεί το οποίο δείχνει μία τοπική όψη του δέντρου XML και την θέση του δρομέα μέσα σε αυτό.

Τα XML στοιχεία δεν εισάγονται ποτέ σιωπηρά εισαχθεί στο έγγραφο. Ωστόσο, μια κοινή δράση στην επεξεργασία εγγράφων, όπως τα αρχεία XML είναι να δημιουργήσετε ένα νέο στοιχείο με το ίδιο όνομα μετά από την τρέχουσα. Η εμφάνιση Author θα εκτελέσει αυτή τη λειτουργία, εάν ο χρήστης πατήσει το πλήκτρο enter δύο φορές (πατώντας μια φορά φέρνει ένα παράθυρο των πιθανών στοιχείων για να προσθέσετε, αν είναι διαθέσιμο).

Η εισαγωγή στοιχείων μπορεί να γίνει μέσω εντολών refactoring της Oxygen xml για την εισαγωγή ενός στοιχείου στην τρέχουσα θέση του δρομέα. Ακόμη και αν οι ετικέτες XML δεν είναι ορατές, μια ένδειξη για ένα άδειο στοιχείο εμφανίζεται πάντα χρησιμοποιώντας το όνομα του εν λόγω στοιχείου.

# 2.7 Graph


Stadium
name
capacity
city
Judge
code
name
surname
gender
date_birth
Games
level
code
gdate
gtime
stadium
sport
judge
Volunteer
at
name
surname
gender
age
Athlites
code
name
weight
surname
country_participation
gender
country_origin
date_birth
trainer_code
height
Sport
code
name
record_name
record_surname
record_date
gender
record
Trainer
code
name
surname
country_origin

Πτυχιακή εργασία της φοιτήτριας Αικατερίνης-Ιωσηφίνα Αρβανιτάκη

Τι πρέπει να υποστηρίζει ένα graphical xml schema.

| | Cxml | *Uml* | *XSD-M* | Altova | Liquid | Oxygen |
|---|---|---|---|---|---|---|
| ***Simple Types*** | | | | | | |
| Xsd Element | √ | √ | √ | √ | √ | √ |
| Xsd Attributes | √ | √ | √ | √ | √ | √ |
| Xsd Restrictions | - | √ | - | - | - | - |
| ***Complex Types*** | | | | | | |
| Xsd Empty | - | - | - | - | √ | - |
| Xsd Elements Only | √ | √ | √ | √ | √ | √ |
| Xsd Text Only | √ | √ | √ | √ | √ | √ |
| Xsd Mixed | √ | √ | √ | √ | √ | |
| Xsd Indicators | - | √ | √ | | √ | √ |
| Xsd <any> | - | √ | - | √ | √ | √ |
| Xsd <anyAttributes> | √ | √ | √ | √ | √ | |
| Xsd Substitution | | √ | | | √ | |
| ***Data* Types** | | | | | | |
| Xsd String | √ | √ | √ | √ | √ | √ |
| Xsd Date | - | √ | √ | √ | √ | √ |
| Xsd Numberic | - | √ | √ | √ | √ | √ |
| Xsd Misc | | | | | | |
| ID | | | | | | |
| IDREF | | | | | | |

Παραπάνω γίνεται η σύγκριση των graphical xml schemes που είδαμε και πιο πάνω, ως προς τι υποστηρίζει το κάθε ένα. Συμπεραίνω ότι το UML και το Liquid είναι τα πιο “αποδοτικά”.

**Στοιχείο:**
Ένα απλό στοιχείο XML το οποίο μπορεί να περιέχει μόνο κείμενο. Δεν περιέχει οποιαδήποτε άλλα στοιχεία ή ιδιότητες. Το κείμενο μπορεί να είναι πολλών διαφορετικών τύπων. Μπορεί να είναι ένα από τα στοιχεία που περιλαμβάνονται στον ορισμό του XML Schema (boolean, αριθμός, ημερομηνία, κλπ.) ή μπορεί να είναι προσαρμοσμένο τύπο που έχετε ορίσει εσείς.
Μπορείτε επίσης να προσθέσετε περιορισμούς (έδρες) σε έναν τύπο δεδομένων για να περιορίσει το περιεχόμενο, ή μπορείτε να ζητήσετε στοιχεία που ταιριάζουν με ένα συγκεκριμένο μοτίβο.

**Χαρακτηριστικά:**
Απλά στοιχεία δεν μπορούν να έχουν χαρακτηριστικά. Αν ένα στοιχείο έχει χαρακτηριστικά θεωρείται ως σύνθετο. Αλλά το χαρακτηριστικό είναι πάντα ίδιο, όπως αναφέρει μια απλή φόρμουλα.

**Περιορισμοί:**
Οι περιορισμοί χρησιμοποιούνται για να καθοριστούν οι αποδεκτές τιμές για XML στοιχεία ή ιδιότητες. Οι περιορισμοί σχετικά με τα στοιχεία XML ονομάζονται πτυχές.

**Κενό:**
Είναι ένα στοιχείο το οποίο δεν έχει περιεχόμενο για τα πάντα. Για να ορίσετε έναν τύπο χωρίς περιεχόμενο, πρέπει να ορίσετε μια λειτουργία που επιτρέπει το περιεχόμενο των δεδομένων, αλλά δεν αναφέρει ορισμένα γεγονότα.

**Στοιχεία μόνο:**
Η elementsOnly είναι ένα στοιχείο της XML, "πρόσωπο" που περιλαμβάνει άλλα αντικείμενα

**Μόνο κείμενο:**
Ο τύπος περιέχει TextOnly μόνο ένα απλό περιεχόμενο (κείμενο και ιδιότητες), προσθέστε ένα στοιχείο έτσι simpleContent γύρω από το περιεχόμενο. Όταν χρησιμοποιείτε ένα απλό περιεχόμενο, πρέπει να καθορίσετε μια επέκταση ή τον περιορισμό μέσα στο στοιχείο simpleContent.

**Μικτή:**
Μια μικτή στοιχείο τύπου μπορεί να περιέχει χαρακτηριστικά, τα στοιχεία και το κείμενο.

**<Any>:**
Η <Any> στοιχείο μας δίνει τη δυνατότητα να επεκτείνει το έγγραφο XML με στοιχεία που δεν προσδιορίζονται από το σχήμα.<anyAttributes>:Η <anyAttribute> στοιχείο μας δίνει τη δυνατότητα να επεκτείνει το έγγραφο XML με ιδιότητες που δεν προσδιορίζονται από το σχήμα.Υποκατάσταση:Η αντικατάσταση στο σχήμα της XML, ένα στοιχείο που μπορεί να αποκαταστήσει ένα άλλο στοιχείο. Σχήματα με την XML, ένα στοιχείο που μπορεί να αντικαταστήσει ένα άλλο στοιχείο.String:Η συμβολοσειρά είναι οι τύποι δεδομένων που

χρησιμοποιούνται για τις τιμές που περιέχουν ακολουθίες χαρακτήρων.Ημερομηνία:Η Ημερομηνία τύπος δεδομένων χρησιμοποιείται για την τιμή που περιέχει ημερομηνία. Για να καθορίσει μια ημερομηνία είναι η ακόλουθη μορφή "EEEE-MM - DD", όπου XXXX δείχνει το έτος, MM δείχνει ο μήνας, HH, δηλώνει την ημέρα. Όλα τα στοιχεία που απαιτούνται.

**Numberic:**

Η numberic είναι δεκαδικό τύπους δεδομένων που χρησιμοποιούνται για αριθμητικές τιμές.Διάφορα:Το Διάφορα είναι τα άλλα είδη των δεδομένων, όπως Boolean, επίπλευσης, διπλό. Ο τύπος boolean έχει οριστεί μια τιμή για να είναι αληθείς ή ψευδείς. Ο τύπος float ορίζει μια σταθερή τιμή.Id:Το id είναι προαιρετική για να γράψει σε ένα σχήμα XML, το οποίο καθορίζει ένα μοναδικό αναγνωστικό κοινό για το στοιχείο.

**IdRef:**

Η idRef είναι η αναφορά σε ένα αναγνωριστικό πεδίου.

# ΚΕΦΑΛΑΙΟ 3

## Binding between XML Schema and Java Classes

"Σύνδεση" μεταξύ σχημάτων XML και Java κλάσεων

## Εισαγωγή

[5][6] Η αρχιτεκτονική Java για Binding XML (JAXB) παρέχει έναν γρήγορο και βολικό τρόπο για να δεσμεύσει μεταξύ τους σχήματα XML και Java παραστάσεις, καθιστώντας πιο εύκολη για προγραμματιστές Java την ενσωμάτωση δεδομένων XML και τις λειτουργίες επεξεργασίας σε εφαρμογές Java. Ως μέρος αυτής της διαδικασίας, η JAXB παρέχει τις μεθόδους για unmarshalling [1] εγγράφων XML σε δέντρα περιεχομένου Java, και στη συνέχεια γίνεται marshalling [2] δέντρα περιεχομένου Java πίσω σε XML έγγραφα. Η JAXB παρέχει επίσης έναν τρόπο για τη δημιουργία σχήματος XML από αντικειμένα Java.

Οι τεχνολογίες XML και Java αναγνωρίζονται ως ιδανικά δομικά στοιχεία για την ανάπτυξη υπηρεσιών και εφαρμογών Web και εφαρμογών που έχουν πρόσβαση στο διαδίκτυο. Η νέα Java API που ονομάζεται JAXB μπορεί να καταστήσει ευκολότερη την πρόσβαση σε έγγραφα XML από εφαρμογές γραμμένες στη γλώσσα προγραμματισμού Java.

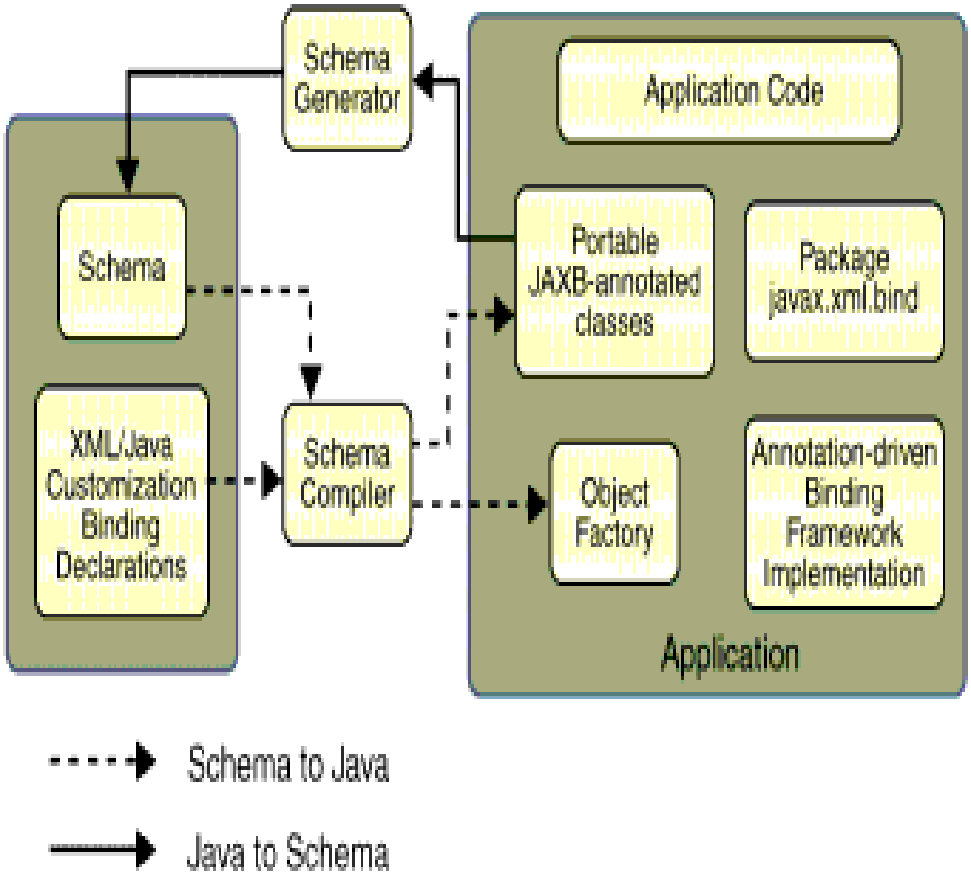
Οι τεχνολογίες XML και Java είναι "συνέταιροι" στο να βοηθάνε προγραμματιστές να ανταλλάσσουν δεδομένα και προγράμματα στο Internet. Αυτό συμβαίνει γιατί η XML έχει αναδειχθεί ως το πρότυπο για την ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων και η τεχνολογία Java παρέχει μια πλατφόρμα για τη δημιουργία φορητών εφαρμογών. Η συνεργασία αυτή είναι ιδιαίτερα σημαντική για τις υπηρεσίες Web, που υπόσχονται στους χρήστες και στους developers εφαρμογών, την λειτουργικότητα των προγραμμάτων οπουδήποτε στο Διαδίκτυο.

Η JAXB επιτρέπει στους Java προγραμματιστές να έχουν πρόσβαση και να επεξεργάζονται δεδομένα XML χωρίς να χρειάζεται να γνωρίζουν XML ή πως να επεξεργάζονται XML έγγραφα.
Η JAXB απλοποιεί την πρόσβαση σε ένα έγγραφο XML από ένα πρόγραμμα Java, παρουσιάζοντας το έγγραφο XML στο πρόγραμμα σε μορφή Java. Το πρώτο βήμα σε αυτή τη διαδικασία είναι να συνδεθεί(binding the schema) το σχήμα σε ένα σύνολο από κλάσεις Java που αναπαριστούν το σχήμα.

Binding a schema σημαίνει δημιουργία ενός συνόλου κλάσεων Java που αναπαριστούν το σχήμα. Όλες οι εφαρμογές JAXB παρέχουν ένα εργαλείο που ονομάζεται μεταγλωττιστής binding για να κάνει binding ένα σχήμα.

## 3.1 Αρχιτεκτονική Επισκόπηση

Σχήμα 2-1 δείχνει τα στοιχεία που συνθέτουν μια εφαρμογή JAXB.




Μια εφαρμογή JAXB αποτελείται από τα ακόλουθα αρχιτεκτονικά στοιχεία:

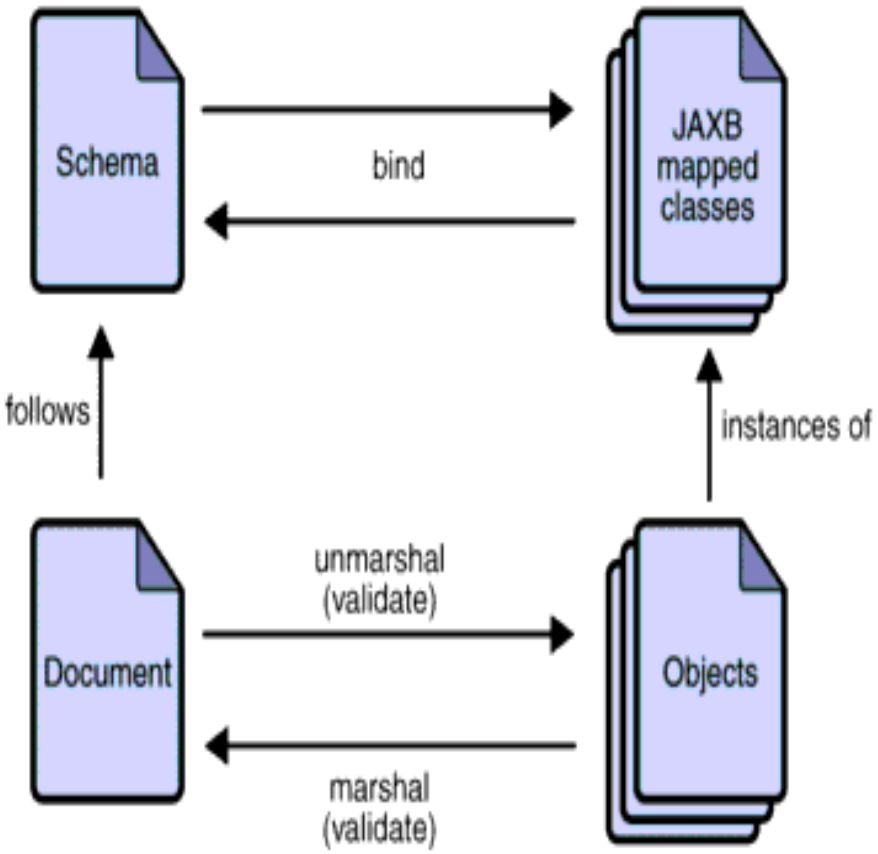
- **schema μεταγλωττιστής** : Συνδέει (binds) ένα xml σχήμα σε μια σειρά από σχήματα που προέρχονται από στοιχεία του προγράμματος. Η σύνδεση αυτή περιγράφεται από μία γλώσσα (binding) βασισμένη στην XML.

- **Γεννήτρια schema** : Χαρτογραφεί ένα σύνολο από στοιχεία προγράμματος σε ένα σχήμα που προκύπτει. Η χαρτογράφηση αυτή περιγράφεται από σχόλια του προγράμματος.

- **binding runtime framework (εκτέλεση πλαισίων binding)**: Παρέχει unmarshalling (ανάγνωση) και marshalling (εγγραφή) εργασιών για την

πρόσβαση, το χειρισμό και την επικύρωση XML περιεχόμενου, χρησιμοποιώντας είτε σχήματα που προκύπτουν είτε προϋπάρχοντα στοιχεία του προγράμματος.

Σχήμα 2-2 δείχνει τι συμβαίνει κατά τη διάρκεια της διαδικασία binding JAXB.




Τα γενικά βήματα της διαδικασία binding JAXB δεδομένων είναι:

1. Δημιουργία κλάσεων. Ένα σχήμα XML χρησιμοποιείται ως πρώτη ύλη από τον μεταγλωττιστή binding JAXB για να δημιουργήσει JAXB κλάσεις με βάση αυτό το σχήμα.

2. Μεταγλωττίστες κλάσεων. Όλες οι κλάσεις που δημιουργούνται, τα αρχεία προέλευσης, και ο κώδικας εφαρμογής, πρέπει να μεταγλωττίζονται.

3. Unmarshal[1]. XML αρχεία που έχουν συνταχθεί σύμφωνα με τους περιορισμούς στο σχήμα πηγή γίνονται unmarshalled[1] από την πλαίσιο binding της JAXB. Σημειώστε ότι η JAXB υποστηρίζει επίσης unmarshalling[1] δεδομένων XML από άλλες πηγές εκτός από τα αρχεία / έγγραφα, όπως τους DOM κόμβους.

4. Δημιουργία δέντρου περιεχόμενου. Η unmarshalling διαδικασία δημιουργεί ένα δέντρο περιεχόμενου από αντικείμενα δεδομένων που τεκμηριώθηκε από τις κλάσεις που δημιουργεί η JAXB. Αυτό το δέντρο περιεχομένου αντιπροσωπεύει τη δομή και το περιεχόμενο των πηγαίων εγγράφων XML.

5. Επικύρωση (προαιρετικά). Η διαδικασία unmarshalling[1] περιλαμβάνει προαιρετικά επικύρωση των πηγαίων εγγράφων XML πριν από τη δημιουργία του δέντρου περιεχομένου. Σημειώστε ότι αν τροποποιήσετε το δέντρο περιεχόμενου στο Βήμα 6, παρακάτω, μπορείτε επίσης να χρησιμοποιήσετε τη λειτουργία Επικύρωσης JAXB για να επικυρώσετε τις αλλαγές πριν γίνει marshalling[2] το περιεχόμενο πίσω σε ένα έγγραφο XML. .

6. Επεξεργασία περιεχομένου. Η εφαρμογή πελάτη μπορεί να τροποποιήσει τα δεδομένα XML που εκπροσωπούνται από το δέντρο περιεχομένου της Java μέσω των διασυνδέσεων που δημιουργούνται από το μεταγλωττιστή δεσμευσής.

7. Marshal[2]. Το επεξεργασμένο δέντρο περιεχόμενου γίνεται marshalled[2] έξω σε ένα ή περισσότερα έγγραφα XML εξόδου. Το περιεχόμενο μπορεί να επικυρωθεί πριν γίνει το marshalling[2].

## 3.2 Παρουσιάζοντας το περιεχόμενο XML

Αυτή η ενότητα περιγράφει το πώς η JAXB παρουσιάζει το περιεχόμενο XML ως αντικείμενα Java. Η JAXB υποστηρίζει την ομαδοποίηση των παραγόμενων κλάσεων σε Java πακέτα.

Ένα πακέτο περιλαμβάνει: Ένα όνομα της κλάσης Java προέρχεται από το όνομα του στοιχείου XML, ή καθορίζεται από ένα δεσμευτικό προσαρμογής. Μια ObjectFactory κλάση είναι ένα εργοστάσιο που χρησιμοποιείται για να επιστρέψει περιπτώσεις bound Java κλάσεων.

### Ορισμοί Απλών Τύπων

Ένα στοιχείο στο σχήμα χρησιμοποιώντας έναν ορισμό απλού τύπου συνδέεται συνήθως με μία ιδιότητα της Java.

Δεδομένου ότι υπάρχουν διάφορα είδη τέτοιων στοιχείων του σχήματος, τα παρακάτω χαρακτηριστικά Java (κοινή για όλα τα συστατικά) περιλαμβάνουν :

- Τύπου Βάση
- Τύπου Συλλογή , εάν υπάρχει
- Κατηγόρημα

Τα υπόλοιπα χαρακτηριστικά Java καθορίζονται στο στοιχείο σχήμα χρησιμοποιώντας το απλό ορισμό του τύπου.

Πίνακας 2-1 JAXB Χαρτογράφηση του XML Schema Built-in Τύποι Δεδομένων

| XML Schema Type | Java Data Type |
|---|---|
| xsd:string | java.lang.String |
| xsd:integer | java.math.BigInteger |
| xsd:int | int |
| xsd.long | long |
| xsd:short | short |
| xsd:decimal | java.math.BigDecimal |
| xsd:double | double |
| xsd:float | float |
| xsd:boolean | boolean |
| xsd:byte | byte |
| xsd:QName | javax.xml.namespace.QName |
| xsd:dateTime | javax.xml.datatype.XMLGregorianCalendar |
| xsd:base64Binary | byte[] |
| xsd:hexBinary | byte[] |
| xsd:unsignedInt | long |
| xsd:unsignedShort | int |
| xsd:unsignedByte | short |
| xsd:time | javax.xml.datatype.XMLGregorianCalendar |
| xsd:date | javax.xml.datatype.XMLGregorianCalendar |
| xsd:g | javax.xml.datatype.XMLGregorianCalendar |
| xsd:anySimpleType | java.lang.Object |
| xsd:anySimpleType | java.lang.String |
| xsd:duration | javax.xml.datatype.Duration |
| xsd:NOTATION | javax.xml.namespace.QName |

### Από κώδικα Java σε σχήμα

Πίνακας 2-2 δείχνει τη χαρτογράφηση προεπιλεγμένων κλάσεων Java σε XML τύπους δεδομένων.

| **Java Class** | **XML Data Type** |
|---|---|
| java.lang.String | xs:string |
| java.math.BigInteger | xs:integer |
| java.math.BigDecimal | xs:decimal |
| java.util.Calendar | xs:dateTime |
| java.util.Date | xs:dateTime |
| javax.xml.namespace.QName | xs:QName |
| java.net.URI | xs:string |
| javax.xml.datatype.XMLGregorianCalendar | xs:anySimpleType |
| javax.xml.datatype.Duration | xs:duration |
| java.lang.Object | xs:anyType |
| java.awt.Image | xs:base64Binary |
| javax.activation.DataHandler | xs:base64Binary |
| javax.xml.transform.Source | xs:base64Binary |
| java.util.UUID | xs:string |

## Προσαρμογή JAXB Bindings

### Από σχήμα σε κώδικα Java

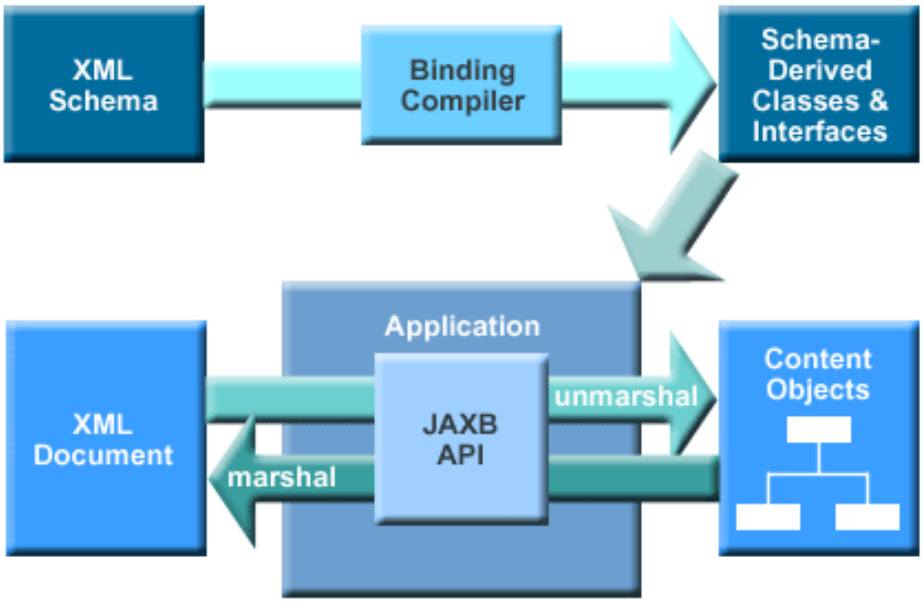
Προσαρμοσμένες δηλώσεις JAXB δεσμεύσεων που σας επιτρέπουν να προσαρμόσετε τις JAXB κλάσεις που δημιουργούνται από την JAXB, πέρα από τα ειδικούς XML περιορισμούς σε ένα σχήμα XML ώστε να συμπεριλάβει συγκεκριμένες Java βελτιώσεις όπως χαρτογραφήσεις κλάσεων και αντιστοιχίσεις ονομάτων του πακέτου.

Η JAXB παρέχει δύο τρόπους για να προσαρμόσετε ένα σχήμα XML:

-Ως ενσωματωμένα σχόλια σε ένα πηγαίο XML σχήμα.

-Ως δηλώσεις σε ένα προσαρμοσμένο εξωτερικό αρχείο δεσμεύσεων που διοχετεύεται στο μεταγλωττιστή binding JAXB.

Το παρακάτω, διάγραμμα από την Java Αρχιτεκτονική για  XML Binding API, παρουσιάζει τις διαδικασίες τόσο για την πρόσβαση όσο και για τη δημιουργία εγγράφων XML από τις εφαρμογές Java.



**Στον παρακάτω πίνακα υπάρχουν παραδείγματα Binding από XML Schema σε Java classes.**

| XML Schema | JAXB Binding |
|---|---|
| `<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">` | |
| `<xsd:element name="purchaseOrder" type="PurchaseOrderType"/>` | PurchaseOrder.java |
| `<xsd:element name="comment" type="xsd:string"/>` | Comment.java |
| `<xsd:complexType name="PurchaseOrderType">`<br>` <xsd:sequence>`<br>` <xsd:element name="shipTo" type="USAddress"/>`<br>` <xsd:element name="billTo" type="USAddress"/>`<br>` <xsd:element ref="comment" minOccurs="0"/>`<br>` <xsd:element name="items" type="Items"/>`<br>` </xsd:sequence>`<br>` <xsd:attribute name="orderDate" type="xsd:date"/>`<br>`</xsd:complexType>` | PurchaseOrderType.java |
| `<xsd:complexType name="USAddress">`<br>` <xsd:sequence>`<br>` <xsd:element name="name" type="xsd:string"/>`<br>` <xsd:element name="street" type="xsd:string"/>`<br>` <xsd:element name="city" type="xsd:string"/>`<br>` <xsd:element name="state" type="xsd:string"/>`<br>` <xsd:element name="zip" type="xsd:decimal"/>`<br>` </xsd:sequence>`<br>`<xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>`<br>`</xsd:complexType>` | USAddress.java |
| `<xsd:complexType name="Items">`<br>` <xsd:sequence>`<br>` <xsd:element name="item" minOccurs="1"`<br>`maxOccurs="unbounded">` | Items.java |
| `<xsd:complexType>`<br>` <xsd:sequence>`<br>` <xsd:element name="productName" type="xsd:string"/>`<br>` <xsd:element name="quantity">`<br>` <xsd:simpleType>`<br>` <xsd:restriction base="xsd:positiveInteger">`<br>` <xsd:maxExclusive value="100"/>`<br>` </xsd:restriction>`<br>` </xsd:simpleType>`<br>` </xsd:element>`<br>` <xsd:element name="USPrice" type="xsd:decimal"/>`<br>` <xsd:element ref="comment" minOccurs="0"/>` | Items.ItemType |

| | |
|---|---|
| <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/><br></xsd:sequence><br><xsd:attribute name="partNum" type="SKU" use="required"/><br></xsd:complexType> | |
| </xsd:element><br></xsd:sequence><br></xsd:complexType> | |
| <!-- Stock Keeping Unit, a code for identifying products --> | |
| <xsd:simpleType name="SKU"><br><xsd:restriction base="xsd:string"><br><xsd:pattern value="\d{3}-[A-Z]{2}"/><br></xsd:restriction><br></xsd:simpleType> | |
| </xsd:schema> | |

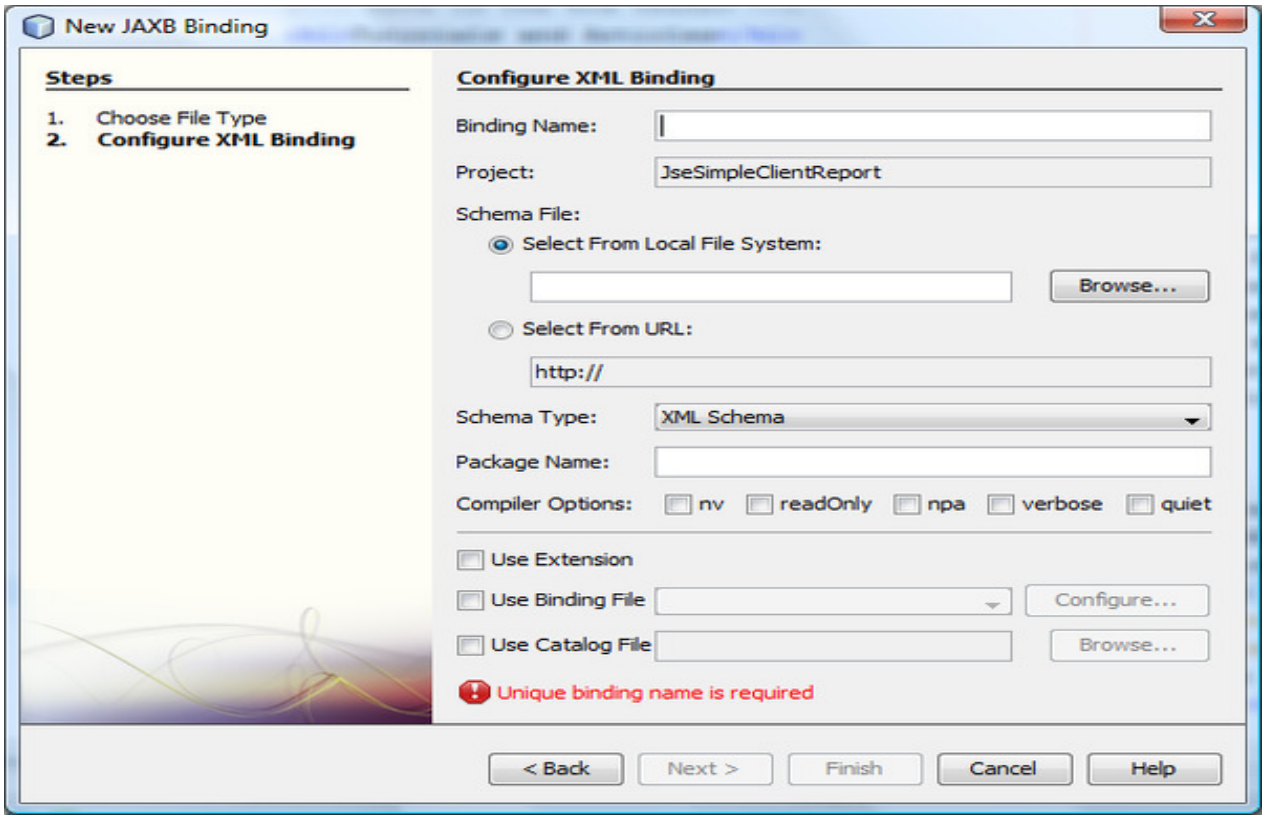
Στην πράξη :

Η τεχνολογία JAXB δεν μπορεί να “σταθεί” μόνη της, δηλαδή να την χρησιμοποιήσουμε χωρίς την ενσωμάτωση της σε κάποιο άλλο πρόγραμμα. Έτσι με την βοήθεια του προγράμματος NetBeans και της Java χρησιμοποιούμε την JAXB. Θα δούμε πως μπορούμε έχοντας αρχεία XML Schema στη διάθεση μας, συγκεκριμένα υποστηρίζονται οι παρακάτω τύποι αρχείων XML Schema :

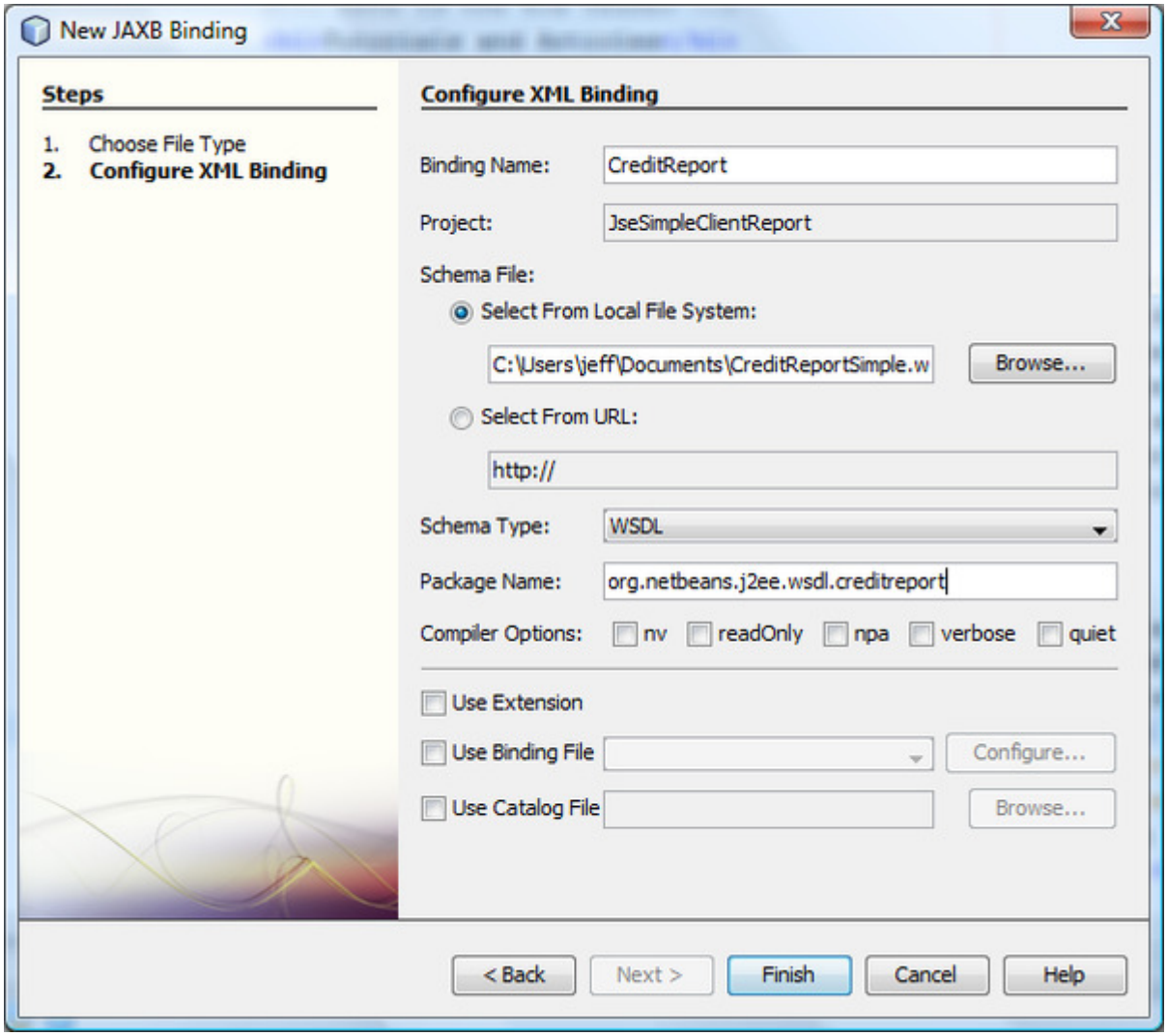
- XML Schema
- XML DTD
- WSDL - Web Service Definition Language. Η γλώσσα XML Schema για τον καθορισμό SOAP-based υπηρεσίες web.

Λοιπόν από τα αρχεία XML Schema μπορούμε να έχουμε Java κλάσεις.

Στο NetBeans, κάνοντας δημιουργία ενός καινούργιου JAXB Binding θα έχουμε το παρακάτω :




Παρακάτω είναι ένα παράδειγμα όπου έχουμε επιλέξει ένα αρχείο XML Schema τύπου WDSL και το ενσωματώνει στο υπάρχων Project JseSimpleClientReport.




Έτσι με το Finish δημιούργουνται οι αντίστοιχες Java classes.

[1] Unmarshlling : παρέχει σε μια εφαρμογή πελάτη τη δυνατότητα να μετατρέπουν τα δεδομένα XML σε αντικείμενα Java που προέρχονται από την JAXB.

[2] marshalling : είναι η διαδικασία μετατροπής της αναπαράστασης στη μνήμη ενός αντικειμένου σε μια μορφή δεδομένων κατάλληλη για αποθήκευση ή μετάδοση, και χρησιμοποιείται συνήθως όταν τα δεδομένα πρέπει να μεταφέρονται μεταξύ διαφόρων τμημάτων ενός προγράμματος  ή από το ένα πρόγραμμα στο άλλο.

# ΚΕΦΑΛΑΙΟ 4

# Παρουσίαση της Εφαρμογής μου.

## Εισαγωγή

Το NetBeans IDE είναι ένα περιβαλλοντικό ανάπτυγμα IDE - ένα εργαλείο στους προγραμματιστές για να γράψουν, να κάνουν compile, debug και να αναπτύξουν προγράμματα. Είναι γραμμένο σε Java - αλλά μπορεί να υποστηρίξει όλες τις γλώσσες προγραμματισμού. Υπάρχει επίσης ένας μεγάλος αριθμός υπομονάδων (modules) που βοηθάνε στην επέκταση της λειτουργικότητας του NetBeans IDE. Το NetBeans IDE είναι ένα ελεύθερο προϊόν δίχως περιορισμούς στον τρόπο χρησιμοποίησής του.

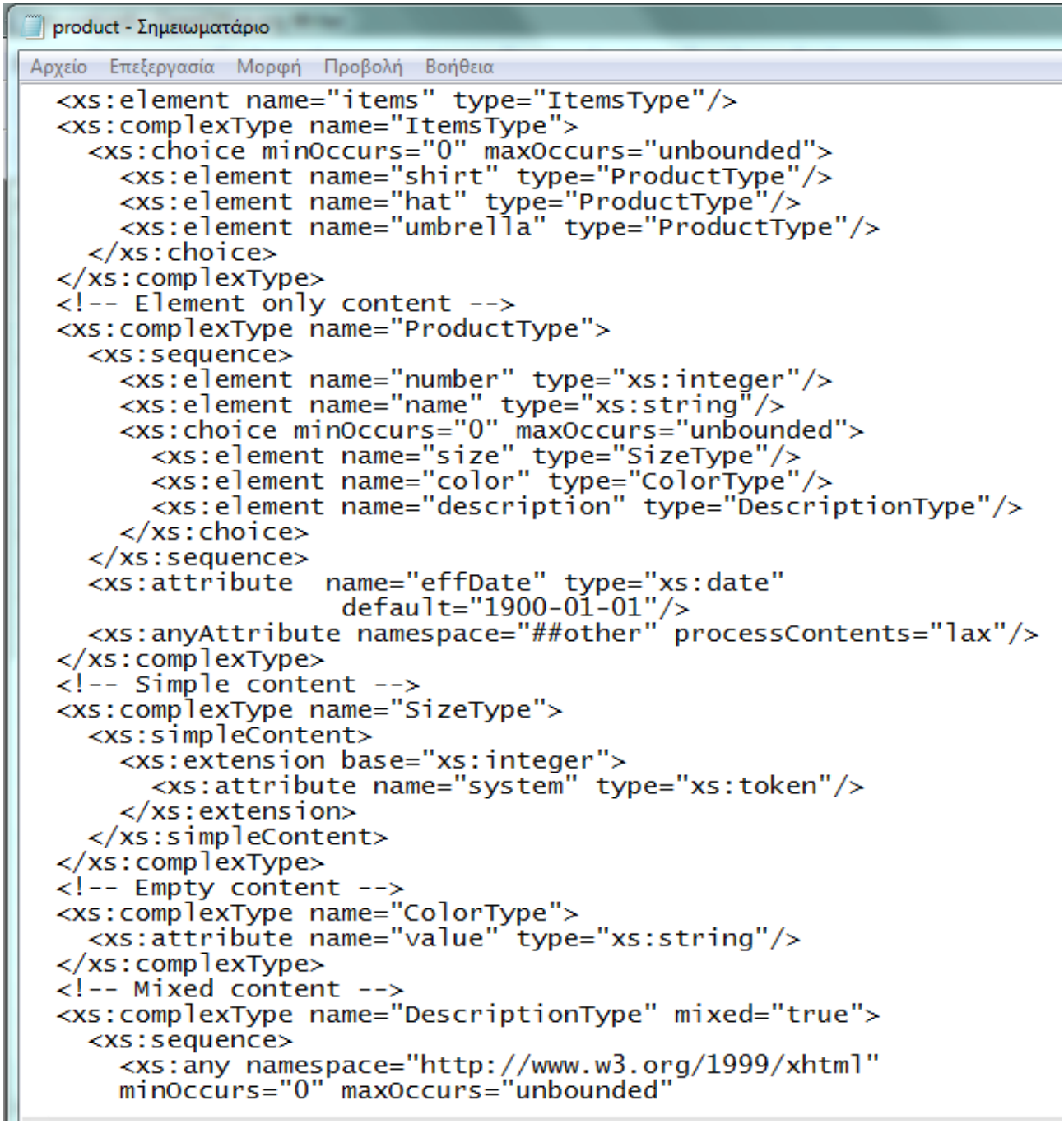
Στην εφαρμογή μου, με την χρήση της τεχνολογίας JAXB, αρχικά μετατρέπω το XML Schema σε Java classes και μετά με την χρήση του γραφικού περιβάλλοντος του NetBeans δείχνω στα στοιχεία του.

Έχοντας μελετήσει και άλλες εφαρμογές οι οποίες παρουσιάζουν XML Schemes γραφικά. Έτσι και αποφάσισα να κάνω την δική μου εκδοχή αναπαράστασης XML Schemes όπου φαίνονται τα στοιχεία του xsd σε πλαίσια διαφορετικού χρώματος αναλόγως με τον τύπο του στοιχείου και τα στοιχεία-παιδιά σε άλλα παράθυρα ώστε να φαίνεται η δόμηση τους.

# Παρουσίαση

[7][8]Στην εφαρμογή μου με το πρόγραμμα NetBeans IDE 7.0.1 και με την χρήση της τεχνολογίας JAXB η οποία είναι ενσωματωμένη, παρουσιάζω ένα xml schema σε γραφικό περιβάλλον.

Συγκεκριμένα θα παρουσιάσω την εφαρμογή μου με ένα παράδειγμα χρησιμοποιώντας το αρχείο customer.xsd. Το οποίο φαίνεται παρακάτω :



Πτυχιακή εργασία της φοιτήτριας Αικατερίνης-Ιωσηφίνα Αρβανιτάκη

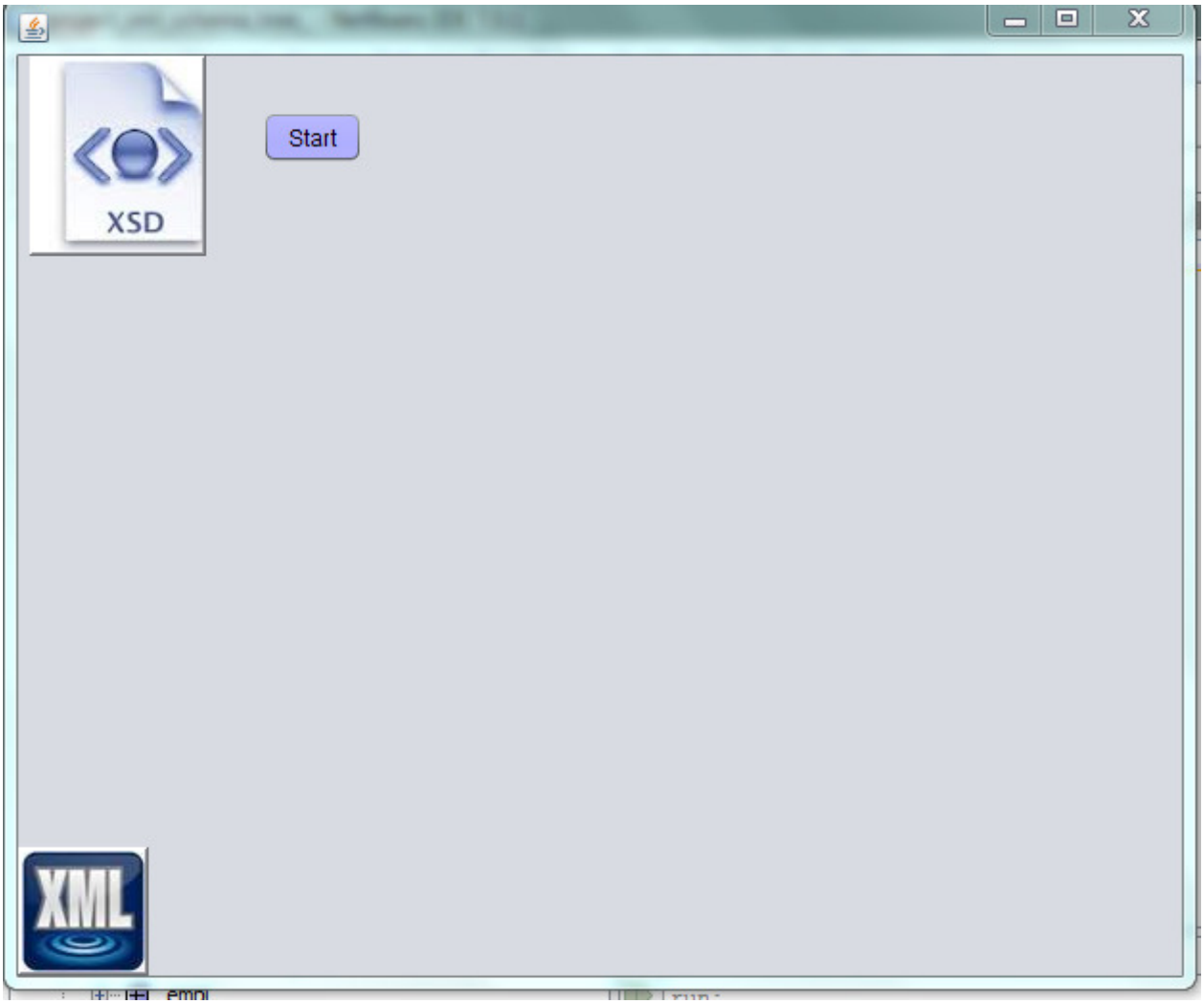
Το οποίο περιέχει 5 complex types το ColorType, το SizeResourceType, το DescriptionType, το ProductType, το ItemsType.
Το ItemsType περιέχει τα στοιχεία(elements) : shirt(ProductType), hat(ProductType), umbrella(ProductType).
Το ColorType περιέχει τα στοιχεία(elements) : value(string).
Το ProductType περιέχει τα στοιχεία(elements) : number(integer), name(string), size(SizeType), color(), description(DescriptionType).

Όπως δείχνω στο κεφάλαιο για την τεχνολογία JAXB, αρχικά μετατρέπω το αρχείο μου product.xsd το οποίο περιέχει ένα xml schema, σε αρχεία java.
Έτσι δημιουργούνται 5 αρχεία java : το ProductType.java, το ColorType.java, το SizeType.java, το DescriptionType.java και το ItemsType.java.

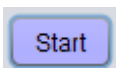
Στο Παράρτημα Α υπάρχουν τα παραπάνω αρχεία Java.

Πρωτού τρέξουμε το Project στην εφαρμογή NetBeans έχουμε την εξής εικόνα :

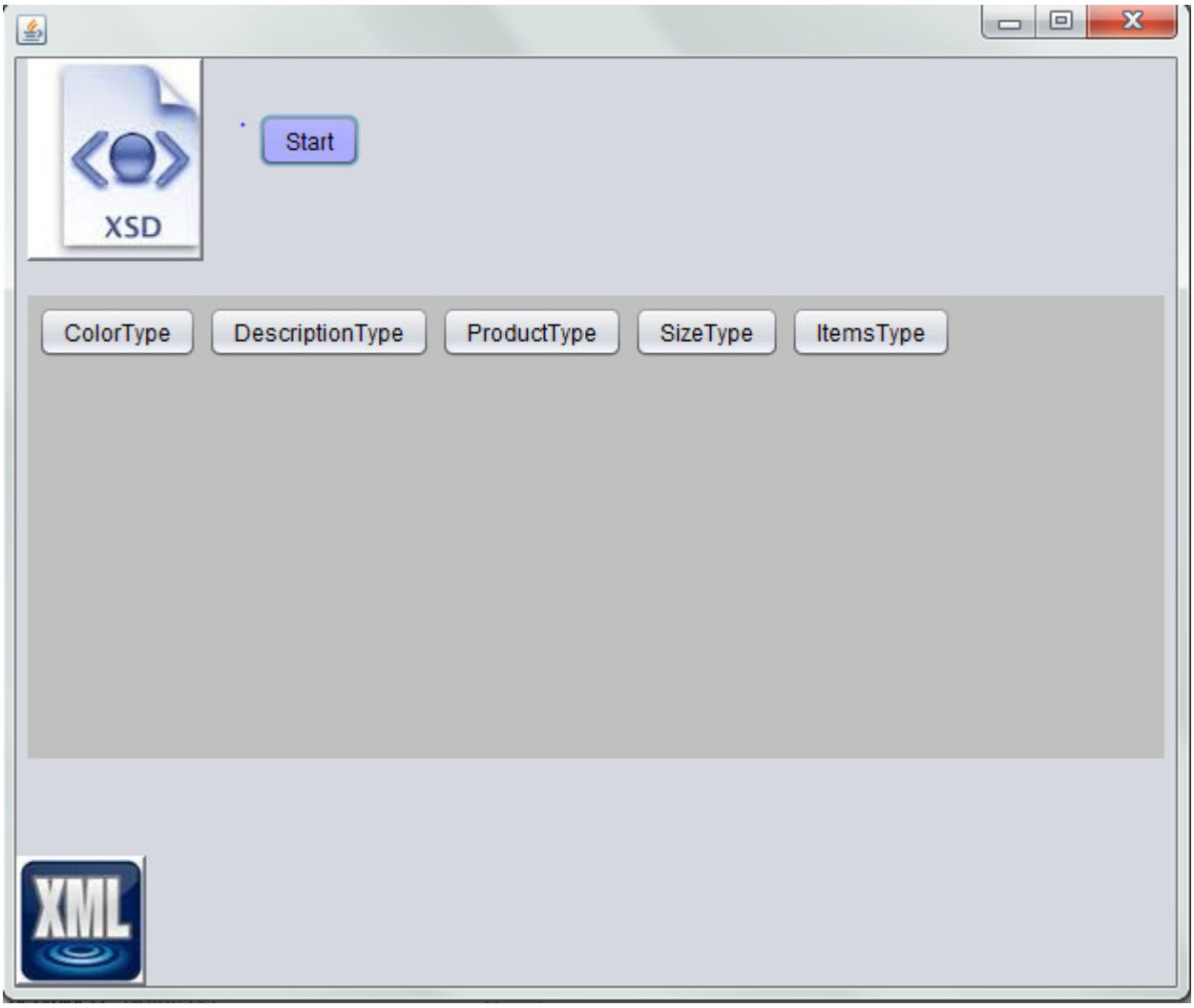
Κάνοντας run το Project εμφανίζεται το εξής παράθυρο :




Στη συνέχεια πατώντας το κουμπί Start :

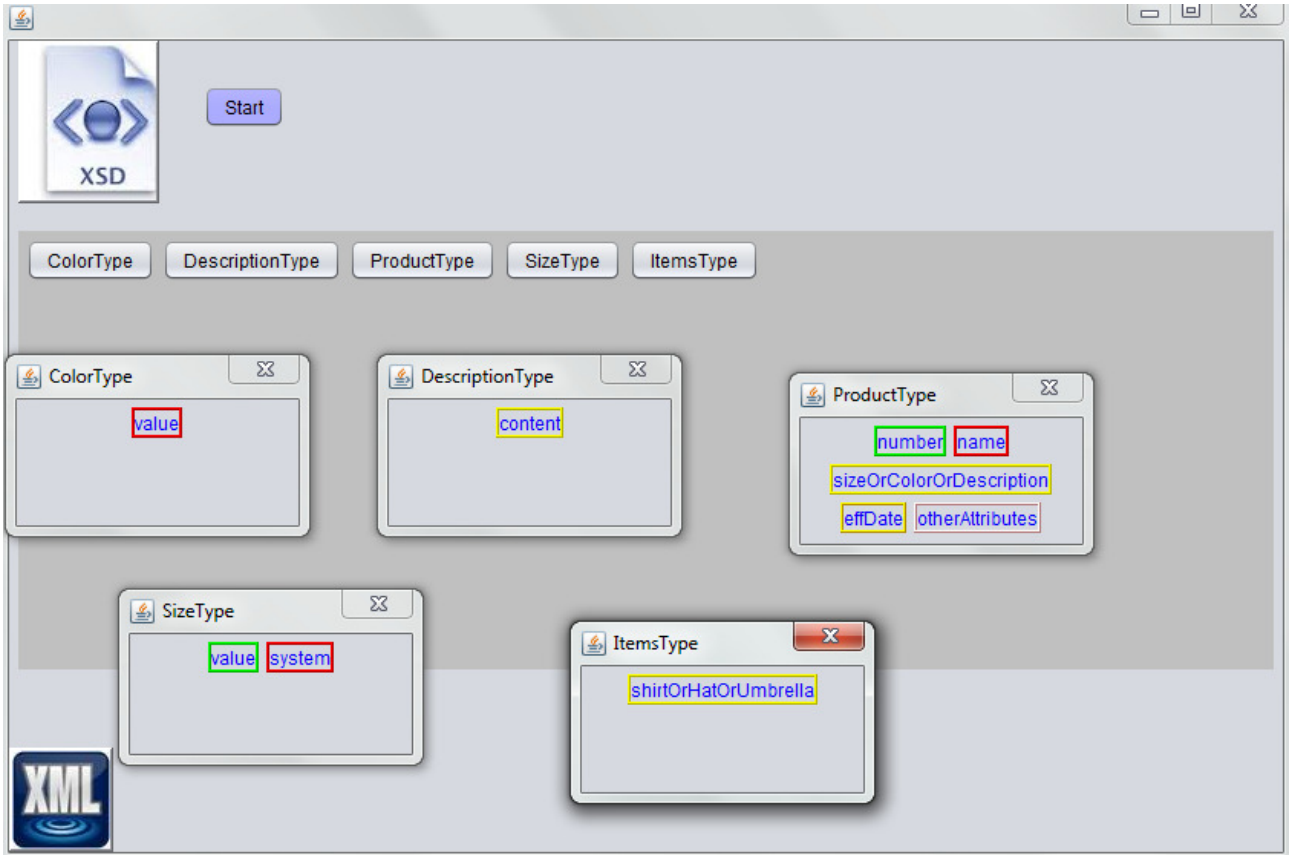



θα έχουμε :




Όπου δημιουργούνται δυναμικά 5 κουμπιά τα οποία αναπαριστούν τα complex types του product.xsd.

Πατώντας τα 5 αυτά κουμπιά δημιουργούνται τα εξής :




Δημιουργούνται 5 ξεχωριστά παράθυρα στα οποία δημιουργούνται μέσα σε labels τα στοιχεία των complex types.
Όπου με κόκκινο πλαίσιο αναπαρίστανται τα στοιχεία τύπου String, με κίτρινο πλαίσιο τα complex στοιχεία,  με πράσινο πλαίσιο τα στοιχεία τύπου Integer.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


1. http://www.w3schools.com/schema/default.asp
2. XML Schema.Eric van der Vlist. Publisher: O'Reilly. First Edition June 2002. ISBN: 0-596-00252-1
3. Beza et al., 2007, Funderburk et. Al., 2002
4. Bekiropoulos et al. (2010)
5. http://www.oracle.com/technetwork/articles/javase/index-140168.html#unmars
6. http://docs.oracle.com
7. http://www.java.com/en/
8. http://netbeans.org/

# ΠΑΡΑΡΤΗΜΑ Α

To SizeType.java:

```
//
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, vhudson-jaxb-ri-2.2-147
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2012.10.21 at 07:57:54 ΕΕΣΤ
//


package generated;

import java.math.BigInteger;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlSchemaType;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.XmlValue;
import javax.xml.bind.annotation.adapters.CollapsedStringAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;


/**
 * <p>Java class for SizeType complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this class.
 *
 * <pre>
 * &lt;complexType name="SizeType">
 *   &lt;simpleContent>
 *     &lt;extension base="&lt;http://www.w3.org/2001/XMLSchema>integer">
 *       &lt;attribute name="system" type="{http://www.w3.org/2001/XMLSchema}token" />
 *     &lt;/extension>
 *   &lt;/simpleContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "SizeType", propOrder = {
    "value"
```

```
})
public class SizeType {

    @XmlValue
    protected BigInteger value;
    @XmlAttribute(name = "system")
    @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
    @XmlSchemaType(name = "token")
    protected String system;

    /**
     * Gets the value of the value property.
     *
     * @return
     *     possible object is
     *     {@link BigInteger }
     *
     */
    public BigInteger getValue() {
        return value;
    }

    /**
     * Sets the value of the value property.
     *
     * @param value
     *     allowed object is
     *     {@link BigInteger }
     *
     */
    public void setValue(BigInteger value) {
        this.value = value;
    }

    /**
     * Gets the value of the system property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getSystem() {
        return system;
    }

    /**
     * Sets the value of the system property.
```

```
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setSystem(String value) {
        this.system = value;
    }

}
```

To ProductType.java:

```
//
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, vhudson-jaxb-ri-2.2-147
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2012.10.21 at 07:57:54 ΕΕΣΤ
//


package generated;

import java.math.BigInteger;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAnyAttribute;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlSchemaType;
import javax.xml.bind.annotation.XmlType;
import javax.xml.datatype.XMLGregorianCalendar;
import javax.xml.namespace.QName;


/**
 * <p>Java class for ProductType complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this class.
 *
```

```
 * <pre>
 * &lt;complexType name="ProductType">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;sequence>
 *         &lt;element name="number" type="{http://www.w3.org/2001/XMLSchema}integer"/>
 *         &lt;element name="name" type="{http://www.w3.org/2001/XMLSchema}string"/>
 *         &lt;choice maxOccurs="unbounded" minOccurs="0">
 *           &lt;element name="size" type="{}SizeType"/>
 *           &lt;element name="color" type="{}ColorType"/>
 *           &lt;element name="description" type="{}DescriptionType"/>
 *         &lt;/choice>
 *       &lt;/sequence>
 *       &lt;attribute name="effDate" type="{http://www.w3.org/2001/XMLSchema}date"
default="1900-01-01" />
 *       &lt;anyAttribute processContents='lax' namespace='##other'/>
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "ProductType", propOrder = {
    "number",
    "name",
    "sizeOrColorOrDescription"
})
public class ProductType {

    @XmlElement(required = true)
    protected BigInteger number;
    @XmlElement(required = true)
    protected String name;
    @XmlElements({
        @XmlElement(name = "size", type = SizeType.class),
        @XmlElement(name = "description", type = DescriptionType.class),
        @XmlElement(name = "color", type = ColorType.class)
    })
    protected List<Object> sizeOrColorOrDescription;
    @XmlAttribute(name = "effDate")
    @XmlSchemaType(name = "date")
    protected XMLGregorianCalendar effDate;
    @XmlAnyAttribute
    private Map<QName, String> otherAttributes = new HashMap<QName, String>();

    /**
```

```
 * Gets the value of the number property.
 *
 * @return
 *     possible object is
 *     {@link BigInteger }
 *
 */
public BigInteger getNumber() {
    return number;
}

/**
 * Sets the value of the number property.
 *
 * @param value
 *     allowed object is
 *     {@link BigInteger }
 *
 */
public void setNumber(BigInteger value) {
    this.number = value;
}

/**
 * Gets the value of the name property.
 *
 * @return
 *     possible object is
 *     {@link String }
 *
 */
public String getName() {
    return name;
}

/**
 * Sets the value of the name property.
 *
 * @param value
 *     allowed object is
 *     {@link String }
 *
 */
public void setName(String value) {
    this.name = value;
}

/**
```

```
     * Gets the value of the sizeOrColorOrDescription property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
sizeOrColorOrDescription property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getSizeOrColorOrDescription().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link SizeType }
     * {@link DescriptionType }
     * {@link ColorType }
     *
     *
     */
    public List<Object> getSizeOrColorOrDescription() {
        if (sizeOrColorOrDescription == null) {
            sizeOrColorOrDescription = new ArrayList<Object>();
        }
        return this.sizeOrColorOrDescription;
    }

    /**
     * Gets the value of the effDate property.
     *
     * @return
     *     possible object is
     *     {@link XMLGregorianCalendar }
     *
     */
    public XMLGregorianCalendar getEffDate() {
        return effDate;
    }

    /**
     * Sets the value of the effDate property.
     *
     * @param value
     *     allowed object is
```

```
     *     {@link XMLGregorianCalendar }
     *
     */
    public void setEffDate(XMLGregorianCalendar value) {
        this.effDate = value;
    }

    /**
     * Gets a map that contains attributes that aren't bound to any typed property on this
class.
     *
     * <p>
     * the map is keyed by the name of the attribute and
     * the value is the string value of the attribute.
     *
     * the map returned by this method is live, and you can add new attribute
     * by updating the map directly. Because of this design, there's no setter.
     *
     *
     * @return
     *     always non-null
     */
    public Map<QName, String> getOtherAttributes() {
        return otherAttributes;
    }

}
```

To ItemsType.java:

```
//
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference
Implementation, vhudson-jaxb-ri-2.2-147
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2012.10.21 at 07:57:54 ΕΕΣΤ
//


package generated;

import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.JAXBElement;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementRefs;
import javax.xml.bind.annotation.XmlType;
```

```
/**
 * <p>Java class for ItemsType complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this
class.
 *
 * <pre>
 * &lt;complexType name="ItemsType">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;choice maxOccurs="unbounded" minOccurs="0">
 *         &lt;element name="shirt" type="{}ProductType"/>
 *         &lt;element name="hat" type="{}ProductType"/>
 *         &lt;element name="umbrella" type="{}ProductType"/>
 *       &lt;/choice>
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "ItemsType", propOrder = {
    "shirtOrHatOrUmbrella"
})
public class ItemsType {

    @XmlElementRefs({
        @XmlElementRef(name = "hat", type = JAXBElement.class, required = false),
        @XmlElementRef(name = "umbrella", type = JAXBElement.class, required = false),
        @XmlElementRef(name = "shirt", type = JAXBElement.class, required = false)
    })
    protected List<JAXBElement<ProductType>> shirtOrHatOrUmbrella;

    /**
     * Gets the value of the shirtOrHatOrUmbrella property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the shirtOrHatOrUmbrella
property.
     *
     * <p>
```

```
     * For example, to add a new item, do as follows:
     * <pre>
     *    getShirtOrHatOrUmbrella().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link JAXBElement }{@code <}{@link ProductType }{@code >}
     * {@link JAXBElement }{@code <}{@link ProductType }{@code >}
     * {@link JAXBElement }{@code <}{@link ProductType }{@code >}
     *
     *
     */
    public List<JAXBElement<ProductType>> getShirtOrHatOrUmbrella() {
        if (shirtOrHatOrUmbrella == null) {
            shirtOrHatOrUmbrella = new ArrayList<JAXBElement<ProductType>>();
        }
        return this.shirtOrHatOrUmbrella;
    }

}
```

To DescriptionType.java :

```
//
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference Implementation, vhudson-jaxb-ri-2.2-147
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2012.10.21 at 07:57:54 EEST
//


package generated;

import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAnyElement;
import javax.xml.bind.annotation.XmlMixed;
import javax.xml.bind.annotation.XmlType;
import org.w3c.dom.Element;


/**
 * <p>Java class for DescriptionType complex type.
 *
```

```
 * <p>The following schema fragment specifies the expected content contained within this
class.
 *
 * <pre>
 * &lt;complexType name="DescriptionType">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;sequence>
 *         &lt;any processContents='skip' namespace='http://www.w3.org/1999/xhtml'
maxOccurs="unbounded" minOccurs="0"/>
 *       &lt;/sequence>
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "DescriptionType", propOrder = {
    "content"
})
public class DescriptionType {

    @XmlMixed
    @XmlAnyElement
    protected List<Object> content;

    /**
     * Gets the value of the content property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the content property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *    getContent().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link String }
     * {@link Element }
```

```
     *
     *
     */
    public List<Object> getContent() {
        if (content == null) {
            content = new ArrayList<Object>();
        }
        return this.content;
    }

}
```

To ColorType.java :

```
//
// This file was generated by the JavaTM Architecture for XML Binding(JAXB) Reference
Implementation, vhudson-jaxb-ri-2.2-147
// See <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of the source schema.
// Generated on: 2012.10.21 at 07:57:54 EEST
//


package generated;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;


/**
 * <p>Java class for ColorType complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within this
class.
 *
 * <pre>
 * &lt;complexType name="ColorType">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;attribute name="value" type="{http://www.w3.org/2001/XMLSchema}string" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
```

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "ColorType")
public class ColorType {

    @XmlAttribute(name = "value")
    protected String value;

    /**
     * Gets the value of the value property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getValue() {
        return value;
    }

    /**
     * Sets the value of the value property.
     *
     * @param value
     *     allowed object is
     *     {@link String }
     *
     */
    public void setValue(String value) {
        this.value = value;
    }

}
```