JAXB, XML schemas & Dozer

Introduction to Service Design and Engineering 2013/2014.

Lab session #4

University of Trento

Cristhian Parra: Lab Lectures for IntroSDE-2013/2014
Some slides from LabSDE-2012/2013

Outline

- XML schema: XSD
- Java Annotation
- Introduction to JAXB
- Example: from schema to java representations
- Example: Generate an XML document from an Object Model
- Exercise
- Dozer
- Assignment

XSD: XML Schema Definition (1)

- An XML schema describes the structure of an XML document
- An XML Schema is written in XML
- It is an XML−based alternative to DTD (document type definition - which is yet another set of markups to learn)
- XML Schema is a W3C Recommendation: http://www.w3.org/2001/XMLSchema

XSD: XML Schema Definition (1)

- An XML Schema defines:
  - **elements** that can appear in a document
  - **attributes** that can appear in a document
  - **data types** for elements and attributes
  - which elements are **child** elements
  - the **order** of child elements
  - whether an element is **empty** or can include **text**
  - **default and fixed values** for elements and attributes

Example 1: XSD (1)

- Open the Example 1

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:example="http://www.example.com/Example"
  targetNamespace="http://www.example.com/Example">
  <xsd:element name="person" type="personType"/>
  <xsd:complexType name="personType">
      <xsd:sequence>
          <xsd:element name="firstName" type="xsd:string"       />
          <xsd:element name="lastName"  type="xsd:string"       />
          <xsd:element name="birthDate" type="xsd:date"       />
          <xsd:element name="age"       type="xsd:integer"      />
          <xsd:element name="healthProfile" type="healthProfileType"  />
       </xsd:sequence>
  <xs:attribute name="id" type="xs:integer"/>
  </xsd:complexType>
  <xsd:complexType name="healthProfileType">
      <xsd:sequence>
          <xsd:element name="weight" type="xsd:decimal"/>
          <xsd:element name="height"  type="xsd:decimal"/>
       </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Example 1: XSD (2)

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:example="http://www.example.com/Example"
    targetNamespace="http://www.example.com/Example">
    ...
 </xsd:schema>
```

- **xmlns:xsd="…"** indicates that the elements and data types used in this schema
  - come from the *http://www.w3.org/2001/XMLSchema* namespace
  - should be prefixed with xsd:

XML Schema built-in data types

- The most common built-in data types are:
  - xsd:string
  - xsd:decimal
  - xsd:integer
  - xsd:boolean
  - xsd:date
  - xsd:time
- The complete built-in data type hierarchy
  - http://www.w3.org/TR/xmlschema-2/#built-in-datatypes

Example 1: XSD (3)

- XML Elements

```
<firstName>George R. R.</firstName>
<lastName>Martin</lastName>
<birthDate>1970-06-21</birthDate>
```

- Corresponding XSD definitions

```
<xsd:element name="firstName" type="xsd:string" />
<xsd:element name="lastName"  type="xsd:string" />
  <xsd:element name="birthDate" type="xsd:date"   />
```

Complex Data Types

- A complex element is an XML element that contains other elements and/or attributes.
- There are four kinds of complex elements:
  - empty elements
  - elements that contain only other elements
  - elements that contain only text (and attributes)
  - elements that contain both other elements and text
- For each kind, there are many ways to write it in a XSD document. We see only one way, that is compatible with JAXB

Example 1: XSD (4)

Empty Elements

- XML Elements

```
<person id="12345">
…
</person>
```

- Corresponding XSD definitions

```
…
<xsd:element name="person" type="personType"/>
  <xsd:complexType name="personType">c
  </xsd:complexType>
```

Example 1: XSD (5)

Elements that contain only Elements

- XML Elements

```
<person>
  <firstName>George R. R.</firstName>
  <lastName>Martin</lastName>
  <birthDate>1970-06-21</birthDate>
</person>
```

- Corresponding XSD definitions

```
<xsd:element name="person" type="personType"/>
  <xsd:complexType name="personType">
      <xsd:sequence>
          <xsd:element name="firstName" type="xsd:string"/>
          <xsd:element name="lastName"  type="xsd:string"/>
          <xsd:element name="birthDate" type="xsd:date"/>
       </xsd:sequence>
  </xsd:complexType>
```

Example 2: XSD (6)

Elements that contain only Text and Attributes

- XML Elements

```
<shoesize country="france">35</shoesize>
```

- Corresponding XSD definitions

```
<xsd:element name="shoesize" type="shoesizeType"/>
  <xsd:complexType name="shoesizeType">
      <xsd:simpleContent>
        <xsd:extension base="xsd:integer">
          <xsd:attribute name="country" type="xsd:string"/>
        </xsd:extension>
       </xsd:simpleContent>
  </xsd:complexType>
```

Example 3: XSD (7)

Elements that contain both elements and text

- XML Elements

```
<p>
  <b>Mixed content</b> lets you embed <i>child elements</i>
</p>
```

- Corresponding XSD definitions

```
  <xsd:complexType name="chunkType" mixed="true">
      <xsd:choice maxOccurs="unbounded">
        <xsd:element name="b" type="chunkType"/>
        <xsd:element name="i" type="chunkType"/>
      </xsd:choice>
  </xsd:complexType>
  <xsd:complexType name="TextType" >
      <xsd:sequence>
        <xsd:element name="p" type="chunkType"/>
      </xsd:sequence>
  </xsd:complexType>
```

- **What was new here?**

XSD Indicators

- **Order indicators** are used to define the order of the elements
  - all, choice, sequence
- **Occurrence indicators** are used to define how often an element can occur
  - maxOccurs, minOccurs
- **Group indicators** are used to define related sets of elements.
  - group name, attributeGroup name

Example 4: XSD (8)

Group names

```
<xsd:group name="persongroup">
    <xsd:sequence>
        <xsd:element name="firstName" type="xsd:string"/>
        <xsd:element name="lastName"  type="xsd:string"/>
        <xsd:element name="birthDate" type="xsd:date"/>
     </xsd:sequence>
</xsd:group>
<xsd:element name="person" type="personinfo"/>
<xsd:complexType name="personinfo" >
    <xsd:sequence>
      <xsd:group ref="persongroup"/>
        <xsd:element name="country" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

Example 5: XSD (9)

AttributeGroup name

```
<xsd:attributeGroup name="elementAttrGroup">
    <xsd:attribute name="refURI" type="xsd:anyURI" use="optional">
    </xsd:attribute>
    <xsd:attribute name="id" type="xsd:integer">
    </xsd:attribute>
</xsd:attributeGroup>
<xsd:complexType name="SomeEntity" >
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attributeGroup ref="elementAttrGroup"/>
        </xsd:extension>
     </xsd:simpleContent>
</xsd:complexType>
```

Example 6: XSD (10)

Sustitution

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

```
<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"/>
```

- Valid XMLs

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

or

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

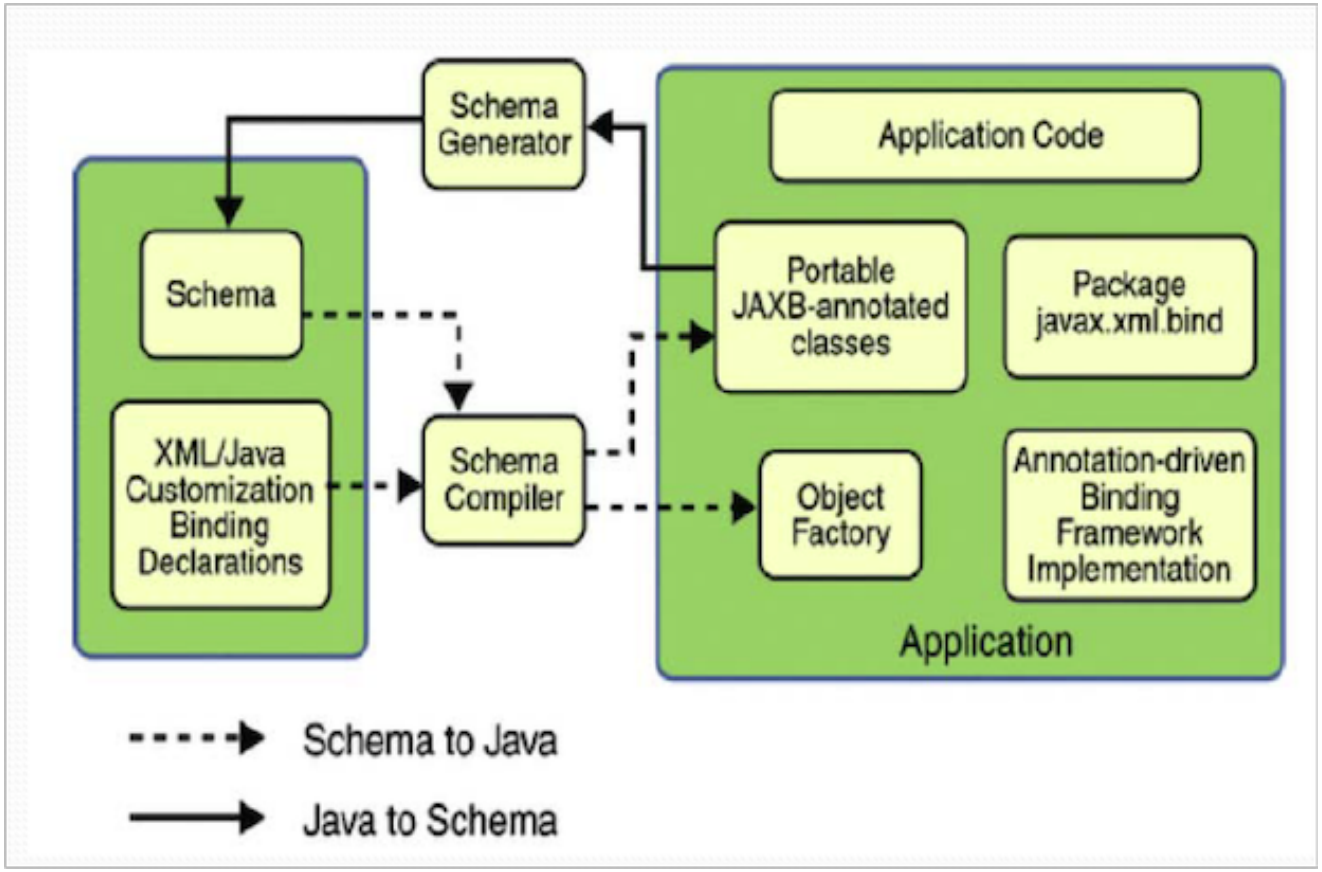
Java Annotations

```
@XmlRootElement // this is a java annotation
@XmlType(name = "", propOrder = { "publisher", "edition", "title", "author" })
public class Catalog {
  private String publisher;
  private String edition;
  private String title;
  private String author;
  ...
  @XmlAttribute
  public String journal;
```

Introduction to JAXB

- JAXB = **J**ava **A**rchitecture for **X**ML **B**inding
- Java standard that defines how Java objects are converted **from** and **to** XML.
- As opposed to XPATH, now we can map XML to a set of Java Classes and restrict our Java program to java objects (not a document tree)
- JAXB Provides two main features:
  - the ability to **marshal** (i.e., convert) Java objects into XML
  - the ability to **un-marshal** XML back into Java objects
- https://jaxb.java.net/

JAXB Architecture



Assignment #1

- Replace the HashMap db in the HealthProfile Reader with a xml file as follows