

XML schemas, Java Annotations, JAXB & Dozer

Introduction to Service Design and Engineering 2013/2014.

Lab session #4

University of Trento

Outline

- XML schema definition (XSD) overview
- Java Annotation minimal example
- Introduction to JAXB
- JAXB Annotations
- Example: XML from/to 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

```
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <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"
        minOccurs="0" maxOccurs="1"/>
      <xsd:element name="healthProfile" type="healthProfileType"
        minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd: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:element name="person" type="personType"/>
</xsd:schema>
```

Example 1: Validate XML against XSD (1)

- Open and XML/XSD validation tool online: <http://www.utilities-online.info/xsdvalidation/#.U10rkGRvj40>
- Copy the content [Example 1 XML Schema](#) and [Example 1 XML instance](#)
- Validate XML againsts XSD

Example 1: XSD (2)

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

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

- Corresponding XSD definitions

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="letter">
<xsd:complexType mixed="true">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="orderid" type="xsd:positiveInteger"/>
    <xsd:element name="shipdate" type="xsd:date"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

- 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>
<xsd:element name="root" type="SomeEntity"/>
```

Exercise 1

- Go to the online validator: <http://www.utilities-online.info/xsdvalidation/#.U10rkGRvj40>
- Copy [Example 5](#) in the XSD Schema.
- Create an XML instance of this schema that is **valid**

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

- Annotations provide data about a program that is not part of the program itself.
- They have no direct effect on the operation of the code they annotate.
- Annotations can be applied to a program's declarations of classes, fields, methods, and other program elements.

```
@XmlElement // 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;
```

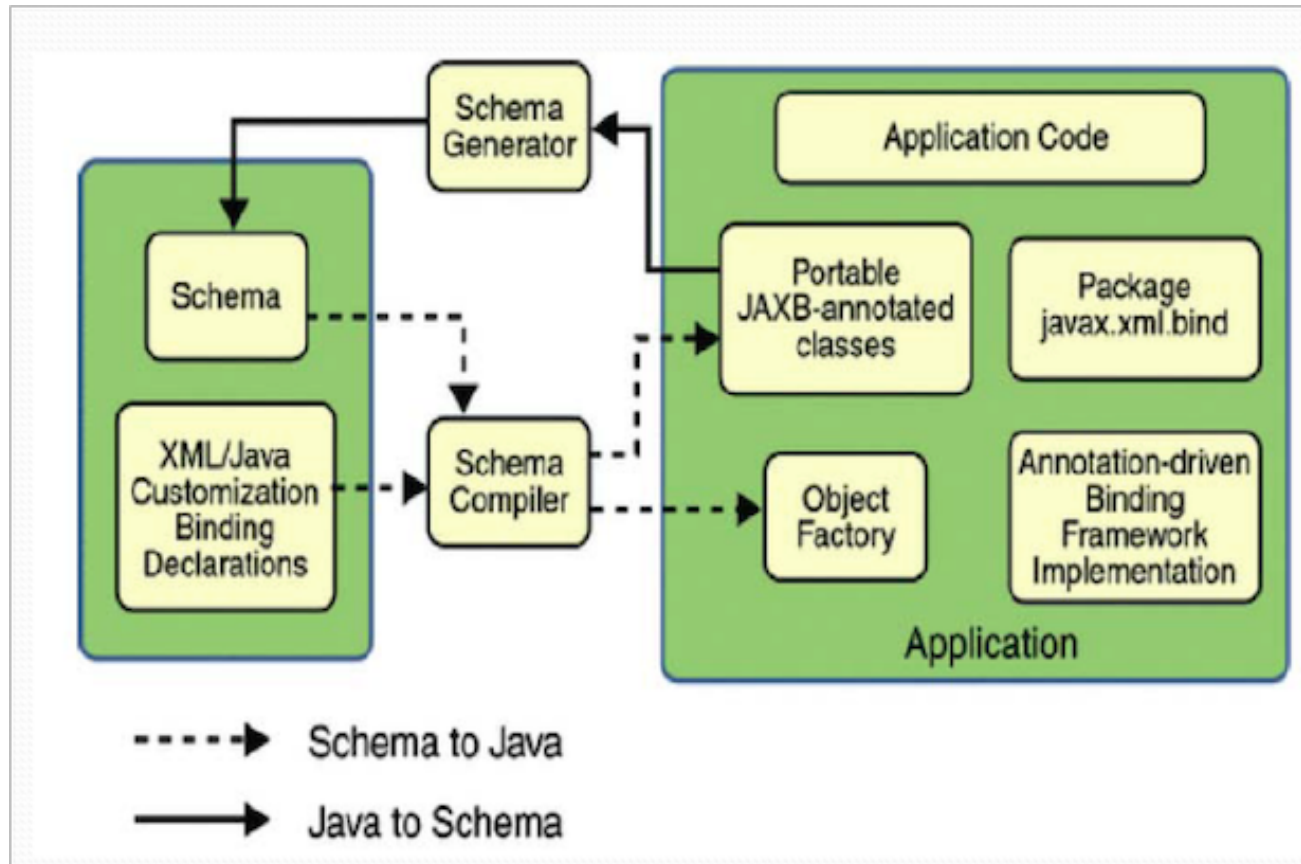
Java Annotations: Uses

- **Information for the compiler:** Annotations can be used by the compiler to detect errors or suppress warnings.
- **Compile-time and deployment-time processing:** Software tools can process annotation information to generate code, XML files, and so forth.
- **Runtime processing:** Some annotations are available to be examined at runtime

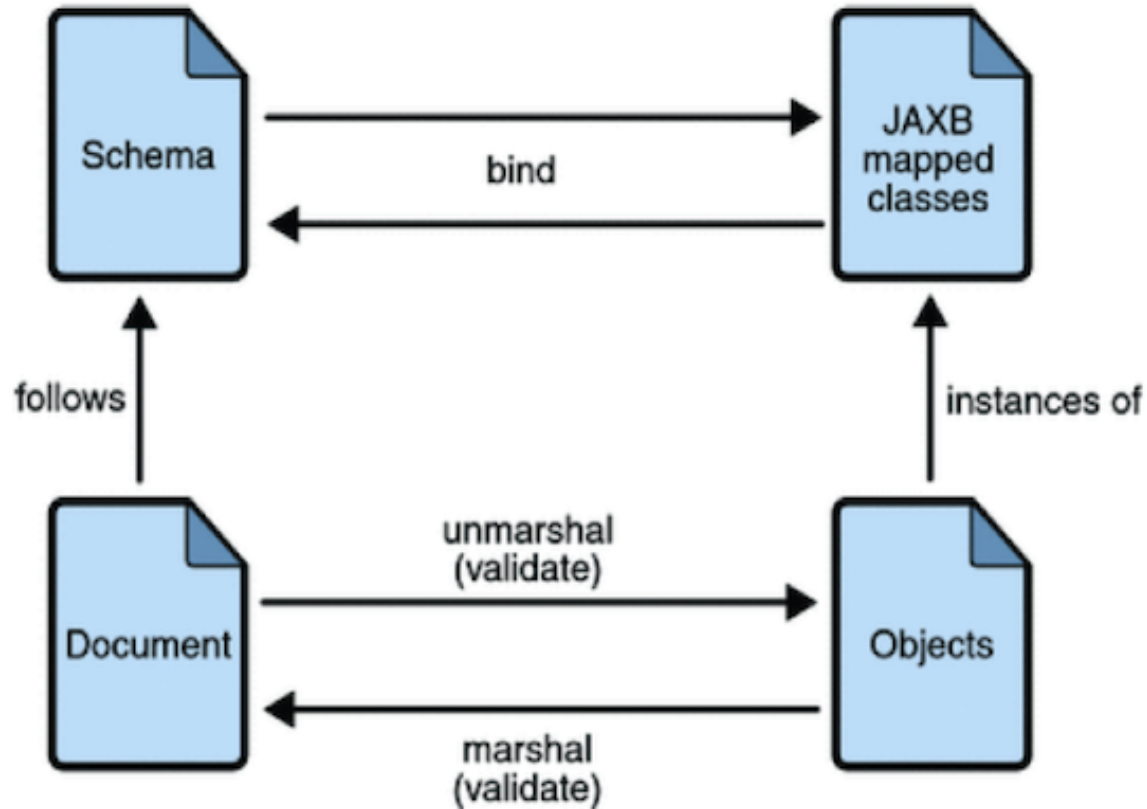
Introduction to JAXB

- 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
- Download from <https://jaxb.java.net/2.2.7/>

JAXB Architecture



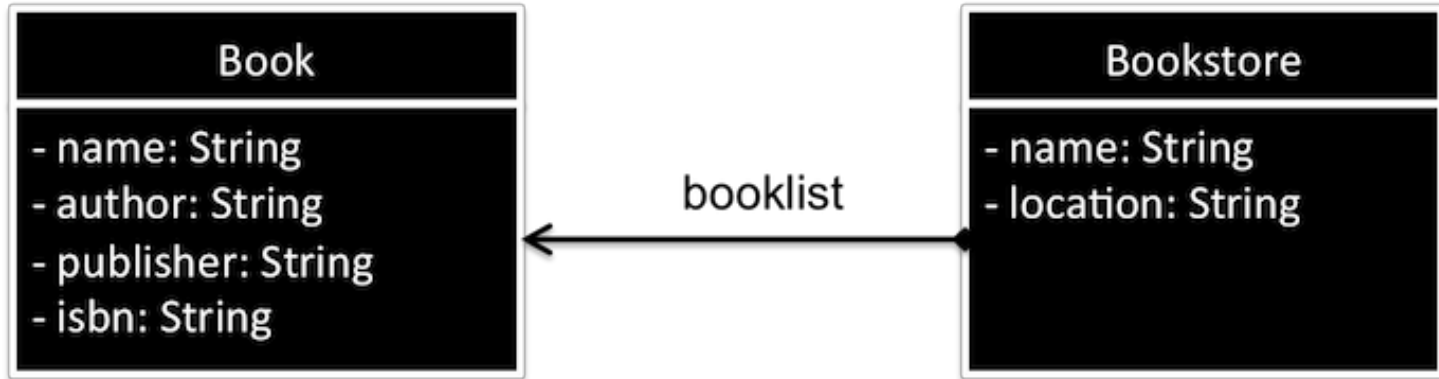
JAXB Binding Process



Source: <http://java.sun.com/javaee/5/docs/tutorial/doc/bnazg.html>

XML From/to Object Model

- Think about the following simple object model



JAXB Annotations

- **@XmlElement(namespace = "namespace")**: defines the root element for an XML tree
- **@XmlType(propOrder = { "field2", "field1", .. })**: allows to define the order in which the fields are written in the XML file
- **@XmlElement(name = "neuName")**: defines the XML element which will be used. Only need to be used if the neuName is different then the JavaBeans Name

Example 6: JAXB Annotations

- Open [Example6-JAXB/src/model/Book.java](#)

```
@XmlRootElement(name = "book")
// If you want you can define the order in which the fields are written
// Optional
@XmlType(propOrder = { "author", "name", "publisher", "isbn" })
public class Book {
    //
    private String name;
    private String author;
    private String publisher;
    private String isbn;
    //
    // If you like the variable name, e.g. "name", you can easily change this
    // name for your XML-Output:
    @XmlElement(name = "title")
    public String getName() {
        return name;
    }
}
```

Example 6: JAXB Annotations

- Open [Example6-JAXB/src/model/Book.java](#)

```
//This statement means that class "Bookstore.java" is the root-element of our example
@XmlRootElement(namespace = "de.vogella.xml.jaxb.model")
public class Bookstore {
    //
    // XmlElementWrapper generates a wrapper element around XML representation
    @XmlElementWrapper(name = "bookList")
    // XmlElement sets the name of the entities
    @XmlElement(name = "book")
    private ArrayList<Book> bookList;
    private String name;
    private String location;
```

Example 6: XML from/to Object Model

- Open [Example6-JAXB/src/model/Book.java](#)

```
public class BookMain {
    // let's put the final result somewhere
    private static final String BOOKSTORE_XML = "./bookstore-jaxb.xml";
    public static void main(String[] args) throws JAXBException, IOException {
        ArrayList<Book> bookList = new ArrayList<Book>();
        // create books
        Book book1 = new Book();
        book1.setIsbn("978-0060554736");
        book1.setName("The Game");
        book1.setAuthor("Neil Strauss");
        book1.setPublisher("Harpercollins");
        bookList.add(book1);
        //
        Book book2 = new Book();
        book2.setIsbn("978-3832180577");
        book2.setName("Feuchtgebiete");
        book2.setAuthor("Charlotte Roche");
        book2.setPublisher("Dumont Buchverlag");
        bookList.add(book2);
        //
        // create bookstore, assigning book
        Bookstore bookstore = new Bookstore();
        bookstore.setName("Fraport Bookstore");
        bookstore.setLocation("Frankfurt Airport");
        bookstore.setBookList(bookList);
        ...
    }
}
```

Example 6: XML from Object Model

```
...
// create JAXB context and instantiate marshaller
JAXBContext context = JAXBContext.newInstance(Bookstore.class);
Marshaller m = context.createMarshaller();
m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
//
// Write to System.out
m.marshal(bookstore, System.out);
//
// Write to File
m.marshal(bookstore, new File(BOOKSTORE_XML));
//
// get variables from our xml file, created before
System.out.println();
System.out.println("Output from our XML File: ");
...
```

Example 6: XML to Object Model

```
...
Unmarshaller um = context.createUnmarshaller();
Bookstore bookstore2 = (Bookstore) um.unmarshal(new FileReader(BOOKSTORE_XML));
ArrayList<Book> list = bookstore2.getBooksList();
for (Book book : list) {
    System.out.println("Book: " + book.getName() + " from "
        + book.getAuthor());
}
}
```

Example 6: XML from/to Object Model

- Enter [Example6-JAXB](#)
- Explore the content of BookMain.java, Book.java and Bookstore.java
- Compile and run the code

```
ant init  
ant compile  
ant execute
```

- What's new in the folder?

Exercise 2: XML from Object Model

- Create a simple Program that creates an xml from a Person model in the following format

```
<person name="Thomas">  
  <age>35</age>  
  <address>Via Malpensada 140</address>  
</person>
```

Example 7: more JAXB

- Enter the directory [Example7-JAXB](#)
- Compile the code and then run

```
javac *.java  
java JavaToXML
```

- Explore catalog.xml

```
java JAXBMarshaller
```

- Explore catalog.xml

```
java JAXBUnMarshaller
```

Exercise 3

- What should you change in [Example7-JAXB](#) to add an **element year** within each article of a journal?

Dozer

- Domain Objects vs Transfer Objects
- I would like to keep domain objects separate from logics that manage the transformation into XML documents
- I propose to define two different packages:
 - entity: here we put pure domain objects
 - transfer: here we put jaxb objects
- We need a mechanism to map domain objects into jaxb objects

Dozer basics

- Dozer is a Java Bean to Java Bean mapper that recursively copies data from one object to another
- Dozer supports mapping between attribute names and between types.
- Standard conversions are provided automatically
- You are allowed to specify custom conversions via XML
- With Dozer, your internal domain objects are not exposed to external presentation layers or to external consumers.
- Dozer maps your domain objects to external APIs calls and vice-versa.
- Dozer can works both with XML and JSON

Dozer Installation

- Download Dozer and extract the archive: <http://dozer.sourceforge.net/>
- Add `${dozer.home}/dist/dozer.jar` to your classpath.
- Add required thirdparty runtime jars to your classpath
<http://dozer.sourceforge.net/dependencies.html>

Assignment #1

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

```
<people>
  <person>
    <firstname>George R. R.</firstname>
    <lastname>Martin</lastname>
    <healthprofile>
      <weight>120</weight>
      <height>1.65</height>
    </healthprofile>
  </person>
  <!-- add more people to the db -->
</people>
```

- **Part 1**

- Extend the example above to include at least 20 people (maybe your friends with fake names)
- Use xpath to implement methods like getWeight and getHeight
- Make a function which prints all people in the list with detail
- A function which accepts name as parameter and print that particular person HealthProfile
- A function which accepts weight and an operator (=, > , <) as parameters and prints people that fulfill that condition.

Assignment #1

- **Part 2**

- Define the XML schema for the example XML document of people.
- Write a java application that does the marshalling and un-marshalling using JAXB.

Assingment Rules

- Before submission make a zip file that includes only
 - All Java source files
 - All XML and XSD files
- Rename the Zip file to: your full name + assignment_no. for example:
cristhian_parra_1.zip
- Submission link: www.dropitto.me/introsde2013
- Password will be given and class and sent to the group
- The assignment is due on 05-Nov (Mid-Night).
- On 06-Nov I'll copy your directory and I use that copy for the evaluation.

For next session

- Install Eclipse: <http://www.eclipse.org/downloads/packages/eclipse-standard-431/keplersr1>
- Install Maven (newly added requirement):
<http://maven.apache.org/download.cgi>
- Prepare yourselves for a long session: we will go till 7pm.
- Stay tuned to the list, will send a list of plugins for eclipse to add later in the week.

References:

- XML Schemas
 - <http://www.w3schools.com/schema/default.asp>
 - <http://www.xfront.com/files/xml-schema.html>
 - Validators:
 - <http://tools.decisionsoft.com/schemaValidate/>
 - <http://www.utilities-online.info/xsdvalidation/#.U10rkGRvj40>
- JAXB
- Dozer

