XML Data Binding

Prof. Dr. Ralf Lämmel
Universität Koblenz-Landau
Software Languages Team
What’s XML binding?
Think of a business programmer who wants to focus on business rules and the object model for the application. Suddenly someone drops X* word on her and she needs to send and receive messages in a XSD-ruled format. How can we make this person happy again?
Example of an XML language: **XBRL**--a language for the electronic communication of business and financial data

**CURRENT ASSETS**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assets Held for Sale</td>
<td>100,000</td>
</tr>
<tr>
<td>Construction in Progress, Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Inventories</td>
<td>100,000</td>
</tr>
<tr>
<td>Other Financial Assets, Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Hedging Instruments, Current (Asset)</td>
<td>100,000</td>
</tr>
<tr>
<td>Current Tax Receivables</td>
<td>100,000</td>
</tr>
<tr>
<td>Trade and Other Receivables, Net,Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Prepayments, Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Cash and Cash Equivalents</td>
<td>100,000</td>
</tr>
<tr>
<td>Other Assets, Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Current Assets Total</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

[http://xbrl.org/](http://xbrl.org/)
<ifrs-gp:AssetsHeldSale contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:AssetsHeldSale>
<ifrs-gp:ConstructionProgressCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:ConstructionProgressCurrent>
<ifrs-gp:Inventories contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:Inventories>
<ifrs-gp:OtherFinancialAssetsCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:OtherFinancialAssetsCurrent>
<ifrs-gp:HedgingInstrumentsCurrentAsset contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:HedgingInstrumentsCurrentAsset>
<ifrs-gp:CurrentTax Receivables contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:CurrentTaxReceivables>
<ifrs-gp:TradeOtherReceivablesNetCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:TradeOtherReceivablesNetCurrent>
<ifrs-gp:PrepaymentsCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:PrepaymentsCurrent>
<ifrs-gp:CashCashEquivalents contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:CashCashEquivalents>
<ifrs-gp:OtherAssetsCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:OtherAssetsCurrent>
<ifrs-gp:AssetsCurrentTotal contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">1000000</ifrs-gp:AssetsCurrentTotal>

http://xbrl.org/
What’s XML data binding?

This is how data is represented externally for storage or exchange.

This is how data is represented for programming.

<xml/>
101 in XML

Think of totaling and cutting salaries for all employees.

XML data binding is when such XML is represented in “business objects” and such operations are implemented in an OO manner.
X-to-O mapping
(as part of XML data binding)

For instance, “xjc” for Technology: JAXB of the Java platform.
Context: X/O/R mapping
The Bermuda Triangle of data processing

In fact, there are further technical spaces.

(C) 2010-2013 Prof. Dr. Ralf Lämmel, Universität Koblenz-Landau (where applicable)
Thanks to Erik Meijer for contributing this slide or some variant thereof.
Preliminaries:
XML processing in languages for XML query / transformation
(as opposed to OO programming languages)
Totaling salaries with XPath

- XPath is an XML query language.
- Queries are composed from query axes.
  - Children, Ancestors, Descendants, ...
- XPath is embedded into Java et al. by APIs.
  - XPath queries are encoded as strings.
- Examples:
  - 
    "//salary"
  - 
    "//manager/salary"

All salary nodes below all manager nodes anywhere
## XPath axes

<table>
<thead>
<tr>
<th>AxisName</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ancestor</td>
<td>Selects all ancestors (parent, grandparent, etc.) of the current node</td>
</tr>
<tr>
<td>ancestor-or-self</td>
<td>Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself</td>
</tr>
<tr>
<td>attribute</td>
<td>Selects all attributes of the current node</td>
</tr>
<tr>
<td>child</td>
<td>Selects all children of the current node</td>
</tr>
<tr>
<td>descendant</td>
<td>Selects all descendants (children, grandchildren, etc.) of the current node</td>
</tr>
<tr>
<td>descendant-or-self</td>
<td>Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself</td>
</tr>
<tr>
<td>following</td>
<td>Selects everything in the document after the closing tag of the current node</td>
</tr>
<tr>
<td>following-sibling</td>
<td>Selects all siblings after the current node</td>
</tr>
<tr>
<td>namespace</td>
<td>Selects all namespace nodes of the current node</td>
</tr>
<tr>
<td>parent</td>
<td>Selects the parent of the current node</td>
</tr>
<tr>
<td>preceding</td>
<td>Selects everything in the document that is before the start tag of the current node</td>
</tr>
<tr>
<td>preceding-sibling</td>
<td>Selects all siblings before the current node</td>
</tr>
<tr>
<td>self</td>
<td>Selects the current node</td>
</tr>
</tbody>
</table>
Cutting salaries with XSLT

- XSLT is an XML transformation language.
- XSLT leverages XPath for node selection.
- XSLT is an XML-based language.
- XSLT is a functional programming language.
Cutting salaries with XSLT

```xml
<xsl:stylesheet>
  <xsl:template match="salary">
    <xsl:copy>
      <xsl:value-of select=". div 2"/>
    </xsl:copy>
  </xsl:template>
  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```
Preliminaries:

XML processing with OO APIs for XML parsing and representation
Options for XML processing in an OO programming language

- APIs for push-based parsers
  - Java’s SAX, ...
- APIs for pull-based parsers
  - .NET’s XmlReader, Java’s StAX, ...
- APIs for in-memory XML trees
  - W3C’s DOM, Java’s JDOM, .NET’s LinqToXml
- Programming languages with XML support
  - VB.NET, ...
The DOM option
(DOM=Document Object Model)

Source: Armstrong: “Working with XML”
What’s DOM?

• An **object model** for XML trees.
• **Central types:**
  – Document
  – Attribute
  – Node
    • Element
    • Text
    • Comment
    • CDATA
• **API segments:**
  – Construction
  – Navigation
  – Modification
http://101companies.org/wiki/
Contribution: dom
The SAX option
(SAX=Simple API for XML)

Source: Armstrong: “Working with XML”
What’s SAX?

- A **framework** for event handling-based XML parsers.
- **Typical events**
  - Open element
  - Close element
  - Find text
  - ...
Demo

http://101companies.org/wiki/
Contribution:sax
XML schemas
XML Schema schemas
(XSDs)
XML languages

- Constrain the set of element *tags* and attributes.
- Constrain the *structure* of elements.
- Other sorts of constraints.
- Thereby XML-based language are defined.
- An XML processor can assume the constraints.
- That is, the processor “knows what to expect”.
- Type systems for valid XML:
  - DTD
  - XML Schema (XSD)
  - Relax NG
XML Schema (XSD)

http://www.w3.org/XML/Schema:
“XML Schemas express shared vocabularies and allow machines to carry out rules made by people. *They provide a means for defining the structure, content and semantics of XML documents.* [...] XML Schema was approved as a W3C Recommendation on 2 May 2001.”
XSD for 101

```xml
<xsl:element name="company">
  <xsl:complexType>
    <xsl:sequence>
      <xsl:element ref="name"/>
      <xsl:element maxOccurs="unbounded"
                   minOccurs="0"
                   ref="department"/>
    </xsl:sequence>
  </xsl:complexType>
</xsl:element>

<xsl:element name="department"> ... </xsl:element>

<xsl:complexType name="employee"> ... </xsl:complexType>
```

XML too

Grammar-like

OO types-like
XML validation

- Well-formed XML as a prerequisite
- Input
  - XML document
  - XML schema
- Output
  - “valid”: document valid w.r.t. schema
  - “invalid” + violations

Compare this to parsing with context-free grammars.
XML validation options

- Validate as XML document is parsed (SAX).
- Validate the XML document built in memory.
- Validate as in-memory tree is serialized.
- ...

(C) 2010-2013 Prof. Dr. Ralf Lämmel, Universität Koblenz-Landau (where applicable)
Demo

http://101companies.org/wiki/
Contribution: sax

See Company.xsd.
See Validator.java.
See unit test for validation.
XSD – schema components

• Element declarations
• Complex type definitions
• Model-group definitions
• Simple type definitions
• Attribute declarations
• Attribute-group definitions
• Redefinitions
• Annotations

Sets of XML trees rooted by a certain element name
Recursive macros with subtyping
Nonrecursive macros without subtyping
Types of leaves in XML trees (both elements and attributes).
Deprecated
Comments and hints for schema processors

Comments and hints for schema processors

Types of leaves in XML trees (both elements and attributes).

Deprecated

Nonrecursive macros without subtyping

Recursive macros with subtyping

Sets of XML trees rooted by a certain element name

(C) 2010-2013 Prof. Dr. Ralf Lämmel, Universität Koblenz-Landau (where applicable)
Sketch of the company schema

<xs:schema ...
  <xs:element name="company"> ... </xs:element>
  <xs:element name="department"> ... </xs:element>
  <xs:complexType name="employee"> ... </xs:complexType>
  <xs:element name="name"> ... </xs:element>
  <xs:element name="address"> ... </xs:element>
  <xs:element name="salary"> ... </xs:element>
</xs:schema>

Exercise: once you have seen the entire schema and completed this lecture, try to answer the following question: Why does it make (some) sense that both element declarations and complex-type definitions are put to work in the sample schema?
Model group compositors

- `<sequence>`: juxtaposition in EBNF
- `<choice>`: `|` in EBNF
- `<all>`: `||` (permutation phrases)
- `minOccurs="0"`:
- `minOccurs="1" maxOccurs=“unbounded”`: +
- `minOccurs="0" maxOccurs=“unbounded”`: *
An employee element has children for name, address, and salary.
In a variation of our preferred schema, a **subunit** (of a department) is either an employee or a department.
A company element has any number of department elements as its children.
Global vs. local

Declaration of a local element declaration

<xs:choice>
  <xs:element name="employee" type="employee" />
  <xs:element ref="department" />
</xs:choice>

Reference to a global element declaration
XSD simple types

• Comparable to primitive types in Java.
• Example:
  
<xs:element name="salary" type="xs:double"/>

• There are predefined simple types in XSD.
• Attributes are of simple types.
• New simple types can be defined by:
  – Restriction
  – Union
  – List
XSD simple type system

Built-in Datatype Hierarchy

- anyType
- anySimpleType
- all complex types

Primitive Types

- duration
- dateTime
- time
- date
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- boolean
- base64Binary
- hexBinary
- float
- double
- anyURI
- QName
- NOTATION
- string
- decimal
- normalizedString
- integer
- token
- nonPositiveInteger
- long
- nonNegativeInteger
- language
- Name
- NMTOKEN
- negativeInteger
- int
- unsignedLong
- positiveInteger
- short
- unsignedInt
- byte
- unsignedShort
- unsignedByte

Derived Types

- derived by restriction
- derived by list
- derived by extension

Built-in primitive types

Built-in derived types
A simple view on XML data binding
O/X type mapping

Maps to

```xml
<element name="point">
  <complexType>
    <sequence>
      <element name="x" type="xs:int"/>
      <element name="y" type="xs:int"/>
    </sequence>
  </complexType>
</element>
```

Maps to

```
public class Point {
  public int x;
  public int y;
}
```
Object model for 101companies System

```java
public class Company {
    private String name;
    private List<Department> depts = new LinkedList<Department>();
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public List<Department> getDepts() { return depts; }
}

public class Department { ... }

public class Employee { ... }
```
XSD for 101companies System

```xml
<xsl:element name="company">
  <xsl:complexType>
    <xsl:sequence>
      <xsl:element ref="name"/>
      <xsl:element maxOccurs="unbounded"
                    minOccurs="0"
                    ref="department"/>
    </xsl:sequence>
  </xsl:complexType>
</xsl:element>

<xsl:element name="department"/>

<xsl:complexType name="employee">
  ...
</xsl:complexType>
```

XML too

Grammar-like

OO types-like
X-to-O mapping

XML types → X-to-O mapping tool → OO types
http://101companies.org/wiki/Contribution:jaxbComposition

For sanity’s sake, let’s look only at Total.java and Cut.java.
XML data binding

- Directions for XML data binding
  - Generate classes from XML schemas.
  - Generate XML schemas from classes.
  - Describe mapping only without generation.

- Motivations for XML data binding
  - Support valid XML output.
  - Hide XML in OO programming.
  - Use XML-based object serialization.
The O/X impedance mismatch
How to map
“s = a:x (b:y+ c:z)+ ”?

```xml
<xs:element name="s">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="a" type="x"/>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="b" type="y" maxOccurs="unbounded"/>
        <xs:element name="c" type="z"/>
      </xs:sequence>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
How to map
“s = a:x (b:y+ c:z)+ ”?

```java
class s {
    x a;
    y[] b;
    z[] c;
}
```

- Grouping of b’s and c’s is lost.
- This may be Ok for read access.
- This is not Ok for round-tripping.
- Occurrence constraints not enforced:
  - Mandatory a, b, c
XML data binding is difficult because XML are parented trees whereas ...
Objects are freewheeling graphs.
Its also difficult because XML trees are node-labeled whereas ...

Thanks to Erik Meijer for contributing this slide or part thereof.
Object graphs are edge-labeled.
Technology: JAXB
Options for schema-derived classes

1. **POJO** with plain fields (and getters/setters):
   - Requires de-/serialization library
   - Object state is “disconnected” from XML trees
   - “XML fidelity” is limited to objects as collections of fields

2. Object types that implement **typed views**:
   - Leverage XML representation:
     • DOM
     • Database
   - Interface provides typed access.
Option chosen by JAXB

• POJO with fields and getters/setters.
• *Annotations* define XSD-related properties.
• “*” and “+” are mapped to *generics*.
• Uses of heterogeneous (weakly typed) containers:
  – Nested composites
  – Mixed context
• An element *factory* is provided.
What's going on here?
public class Cut {
    public static void cut(Company c) {
        for (Department d : c.getDepartment())
            cut(d);
    }
    public static void cut(Department d) {
        cut(d.getManager());
        for (Department s : d.getDepartment())
            cut(s);
        for (Employee e : d.getEmployee())
            cut(e);
    }
    public static void cut(Employee e) {
        e.setSalary(e.getSalary() / 2);
    }
}
Un-/marshaling

```java
public static Company readCompany(File input) throws JAXBException {
    JAXBContext jaxbContext = JAXBContext.newInstance("org.softlang.company");
    Unmarshaller unMarshaller = jaxbContext.createUnmarshaller();
    return (Company)unMarshaller.unmarshal(input);
}
```
Annotations

```java
@XmlElementType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "name",
    "department"
})
@XmlRootElement(name = "company")
public class Company {

    @XmlElement(required = true)
    protected String name;

    protected List<Department> department;

    ...
}
```
Liberal mapping

```java
public class Subunit {
    protected Employee employee;
    protected Department department;
    public Employee getEmployee() { return employee; }
    public void setEmployee(Employee value) { this.employee = value; }
    public Department getDepartment() { return department; }
    public void setDepartment(Department value) { this.department = value; }
}
```

What is (too) liberal here?
XML data binding with **JAXB**

JAXB is an integral part of the Java SDK since Java 6.
Samples on 101

- Contribution: `jaxbComposition`
- Contribution: `jaxbChoice`
- Contribution: `jaxbExtension`
- Contribution: `jaxbSubstitution`
Summary

- XML serves important scenarios.
  - Data exchange and storage
  - Language-independent data model
  - Semi-structured data
- **XML isn’t optimal.**
  - It is complex, in fact, a job-security technology.
  - Fortunately, there are alternatives: JSON, JAXB, ...
  - XML programming could be simpler.
- Manageability of XML data binding varies.
  - Looks seductively simple at first.
  - Becomes more complex eventually.
  - O/X impedance mismatch to be resolved.