Extensible Markup Language (XML), XML Namespaces and XML Schema

BWA Slides #2

XML Technologies

Huge family of technologies and data formats
- XML, XML Schema, XPath, XLink, XPointer, XSLT, XHTML, XML Encryption, SVG, ...

We'll use XML as little as possible
- Configuration files in JEE uses XML
- Marshal some object to XML

Need some understanding, we'll examine
- XML
- XML Namespaces
- XML Schema (and later XHTML)

XML

“Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them”
// W3C Recommendation XML 1.0 (Fifth ed)

[Design goals for XML]

XML can represent any data, also programs like [XSLT]

XML Documents

Syntax of XML Document

Basically

- Unicode
- White space preserved
- Case sensitive
- Must be well formed (matching tags, a tree structure). If not well formed should be rejected it (exception).
- Top level structure: document ::= prolog element Misc*
- Prolog first, really, no whitespace before
- Elements (tags) one or more
- No default rendering style

Element Content

Elements have content
- Except empty element (like <br/>)

Basically
- Other elements (nested)
- Characterdata (non markup text, the "real" data)
- Comments <!-- … --> (also possible outside any element)
- CDATA, escaped markup
- Entity references like &amp; (&)

Attributes

Any element start tag may have attributes

Basically

- Syntax: <element name = value …> … </element>
- Name unique in element
- Value as string "829" or '829' (may not contain "<")
- Space separates attributes
- Ordering doesn't matter

XML Markup Collision

XML is <u>extensible</u> so anyone can create tag sets, "markup vocabulary"

" … documents, containing multiple markup vocabularies, pose problems of recognition and collision. Software modules need to be able to recognize the elements and attributes which they are designed to process, even in the face of "collisions" occurring when markup intended for some other software package uses the same element name or attribute name."
//[Namespaces in XML 1.0 (Third Edition)]

XML Namespaces

Namespaces is a solution to the collision problem

Basically

- An XML namespace is identified by a URI reference (globally unique, possibly meaningless, “nothing there”)
- Prefix element names with namespace
- Namespace and name separated with “:”
- Possibly many namespaces in same document
- URIs long strings, define abbreviation in namespace binding

XML Namespace Binding

"… a namespace binding is declared using a family of reserved attributes. Such an attribute's name must either be xmlns or begin xmlns: [xmlns = XML Namespace]"

- Binding normally inside document root element
- Eliminating use of long strings

```
<!-- URI has no meaning, just a unique URI -->
<c:catalog xmlns:c="http://www.chalmers.se/hajo/cdreg">
    <c:cd id="1">
        ...
    </c:cd>
</c:catalog>
```

XML Schemas

In Java (class based OO)

- An object is an instance of a class (type)
- Type system can check a lot

In XML

- A document is an instance of an schema
- A Schema defines "the type" of the document
- Possible to verify "typecheck" documents

Recap: "Definition: A data object is an XML document if it is well-formed, as defined in this specification. In addition, the XML document is valid if it meets certain further constraints. [i.e. adheres to some schema]"

XML Schema (no ending s)

“Recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document.”
// Wikipedia

- XML Schema written as an XML document!
- XML Schema files uses .xsd suffix (XML Schema Definition)
- XML Schema uses elements from the xsd namespace

```
<xsd:simpleType .... />
```

XML Schema Recommendation

“The purpose of an XML Schema: Structures schema is to define and describe a class of XML documents by using schema components to constrain and document the meaning, usage and relationships of their constituent parts: datatypes, elements and their content and attributes and their values.”
//http://www.w3.org/TR/xmlschema-1/ (Structures)

- Also http://www.w3.org/TR/xmlschema-2/ (Datatypes)
- Known to be very complex many hundred pages, got severe criticism
- Dependencies on other specifications: XML Namespaces, XPath, …

XML Schema Simple Types

Predefined: boolean, string, decimal, double, float, anyURI, QName, hexBinary, base64Binary, duration, date, time, dateTime, gYear, gYearMonth, gMonth, gMonthDay, gDay, and NOTATION

Create own simple type String20 from predefined simple type xsd: string

```
<xsd:simpleType name="String20">
    <xsd:restriction base="xsd:string">
        <xsd:maxLength value="20" />
    </xsd:restriction>
 </xsd:simpleType>
```

XML Schema Complex Types

Creating a complex type "Country" (note leading uppercase). A convention
- Using our String20 type

```
<xsd:complexType name="Country">
    <xsd:sequence>
       <xsd:element name="name" type="String20"/>
       <xsd:element name="population"type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:int" use="required" />
</xsd:complexType>
```

Simple types can't have attributes

Defining an Element

Define element country of type Country (using our type own Country type)

```
<!-- In Schema,define element -->
<xsd:element name="country" type="Country"/>


<!-- In document, use element -->
<country> ... </country>
```

Schema Constrained Document

In XML instance (document) specify the schema to use. Possible for software to check validity

```
 <root xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="...URI to schema..." >
    :
  </root>
```

- xsi prefix elements from XML Schema instance namespace
- xsi:schemaLocation, where to find...

XML Schema and Namespaces

The elements declared in the schema could (should!) belong to a namespace, the "targetNamespace"

Schema can force documents to use qualified elements and attributes
- elementFormDefault="qualified" (or “unqualified”)
- attributeFormDefault="unqualified" (or “qualified”)
- Default values for both: “unqualified"

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:tns="http://www.chalmers.se/hajo/schema/country"
           targetNamespace="http://www.chalmers.
se/hajo/schema/country"
           elementFormDefault="qualified">
```

XML Schema and Name Spaces Examples

[Specification section 3]()

Modular XML Schema's

Possible to modularize schema's (reuse types)
- Use type Country to define type CountryList

```
<xsd:complexType name="CountryList">
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
       <!-- Reuse country element from other schema -->
       <xsd:element ref="c:country"/>
   </xsd:sequence>
</xsd:complexType>
```

We're always interested in modularity, basic principle!

Programmatic Handling of XML

Many (most, all?) languages have API's for XML handling

XML as documents
- SAX, Handled as a stream of chars. Only small parts of document in memory. Can't manipulate document
- DOM (the XML DOM), Converting between text and in memory tree representation. Complete document in memory!

XML from/to classes/objects.
- Converting/Serializing between documents and classes/objects in OO languages.

Java API's for XML

JAXP (javax.xml.*, org.w3c.dom.* org.xml.sax.*)

- SAX, DOM, XPath, XSLT and more ...

JAXB (javax.xml.bind.*)

- Map between XML Document/Schema and object/class
- <u>Used in the background with Web Services,</u> more to come

Non standard

- JDOM, XStream, ...