Processing of structured documents

Spring 2003, Part 3
Helena Ahonen-Myka

XML Namespaces

- An XML document may contain multiple markup vocabularies
- reuse of existing markup, e.g. including HTML markup in some document type
- An XML namespace is a collection of names, identified by a URI reference, which are used in XML documents as element and attribute names

Author A writes a document:

```
<?xml version="1.0"?>
<references>
   <name>Macmillan</name>
   <link href="http://www.mcp.com"/>
   <name>ABC News</name>
   <link href="http://www.abcnews.com"/>
</references>
```


Author B adds some rating....

```
<?xml version="1.0"?>
<references>
   <name>Macmillan</name>
   <link href="http://www.mcp.com"/>
   <rating>5 stars</rating>
   <name>ABC News</name>
   <link href="http://www.abcnews.com"/>
   <rating>3 stars</rating>
</references>
```


Also Author C wants to add some rating...

```
<?xml version="1.0"?>
<references>
   <name>Macmillan</name>
   <link href="http://www.mcp.com"/>
   <rating>G</rating>
   <name>ABC News</name>
   <link href="http://www.abcnews.com"/>
   <rating>PG</rating>
</references>
```


Author D would like to combine the documents...

```
<?xml version="1.0"?>
<references>
   <name>Macmillan</name>
   <link href="http://www.mcp.com"/>
   <rating>5 stars</rating>
   <rating>G</rating>
   <name>ABC News</name>
   <link href="http://www.abcnews.com"/>
   <rating>3 stars</rating>
   <rating>PG</rating>
</references>
```

Which rating? -> different names

```
<?xml version="1.0"?>
<references>
   <name>Macmillan</name>
   <link href="http://www.mcp.com"/>
   <qa-rating>5 stars</qa-rating>
   <pa-rating>G</pa-rating>
   <name>ABC News</name>
   <link href="http://www.abcnews.com"/>
   <qa-rating>3 stars</qa-rating>
   <pa-rating>PG</pa-rating>
</references>
```

7

Namespaces give a disciplined method for naming

```
<?xml version="1.0"?>
<references xmlns:qa="http://joker.com/2000/star-rating"
            xmlns:pa="http://penguin.xmli.com/2000/review"
            xmlns="http://pineapplesoft.com/1999/ref">
   <name>Macmillan</name>
   <link href="http://www.mcp.com"/>
   <qa:rating>5 stars</qa:rating>
   <pa:rating>G</pa:rating>
   ...
</references>
```

8

Namespaces

- xmlns:qa="http://joker.com/2000/star-rating"
  - qa: prefix
  - http://joker.com/2000/star-rating
    - the namespace
    - a unique name (URI guarantees): no need to retrieve anything from the address
- xmlns=" http://pineapplesoft.com/1999/ref">
  - the default namespace
  - elements without prefixes belong to this namespace
    - references, name, link

9

Namespaces

- qa:rating
  - a qualified name (Qname)
- scoping:
  - The namespace is valid for the element where it is declared and all the elements within its content

10

Scoping

```
<?xml version="1.0"?>
<ref:references xmlns:ref="http://pineapplesoft.com/1999/ref">
   <ref:name>Macmillan</ref:name>
   <ref:link href="http://www.mcp.com"/>
   <pa:rating
xmlns:pa="http://penguin.xmli.com/2000/review">G</pa:rating>
   <ref:name>ABC News</ref:name>
   <ref:link href="http://www.abcnews.com"/>
   <qa:rating  xmlns:qa="http://joker.com/2000/star-rating">
        3 stars</qa:rating>
</ref:references>
```

11

Attributes

- if an attribute is
  - **qualified** in an instance (= with namespace prefix), its name is unique in that namespace
  - **unqualified** in an instance, the combination of the attribute name with the element's type and namespace name uniquely identifies the attribute
    - E.g. the "href" attribute above is identified by "link", http://pineapplesoft.com/1999/ref, and "href"
    - note: an element can have only one attribute with the same name

12

Namespaces and DTD

- XML 1.0 DTDs are not namespace-aware
- all the elements and attributes that are in some namespace have to be declared using the corresponding prefix
- for elements with prefix 'pre' :
  - an attribute 'xmlns:pre' has to be declared

13

Namespaces and DTD

```
<?xml version="1.0"?>
<!DOCTYPE ref:references [
<!ELEMENT ref:references
        (ref:name, ref:link, (pa:rating | qa:rating)*)+>
<!ATTLIST ref:references xmlns:ref CDATA #REQUIRED>
<!ELEMENT ref:name  (#PCDATA)>
<!ELEMENT ref:link  EMPTY>
<!ATTLIST ref:link   href  CDATA #REQUIRED>
<!ELEMENT pa:rating (#PCDATA)>
<!ATTLIST pa:rating xmlns:pa CDATA #REQUIRED>
<!ELEMENT qa:rating (#PCDATA)>
<!ATTLIST qa:rating xmlns:qa CDATA #REQUIRED>
]>
```

14

Modularity with DTDs: external and internal subsets

- external and internal subset make up the DTD; internal has higher precedence
- syntax:
  - <!DOCTYPE root-type-name SYSTEM "ex.dtd"
    <!-- external subset in file ex.dtd -->
    [<!-- internal subset may come here -->]>
- internal subset may declare new elements (with attributes) or new attributes for existing elements
- XML Schema is namespace-aware -> more modularity

15

XML Schema

- **DTDs have drawbacks:**
  - they can only define the element structure and attributes
  - they cannot define any database-like constraints for elements:
    - Value (min, max, etc.)
    - Type (integer, string, etc.)
  - DTDs are not written in XML and cannot thus be processed with the same tools as XML documents, XSL(T), etc.
  - difficult to combine different vocabularies (namespaces)
- **XML Schemas**:
  - are written in XML
  - avoid most of the DTD drawbacks

16

XML Schema

- **XML Schema Part 1: Structures:**
  - Element structure definition as with DTD: Elements, attributes, also enhanced ways to control structures
- **XML Schema Part 2: Datatypes:**
  - Primitive datatypes (string, boolean, float, etc.)
  - Derived datatypes from primitive datatypes (time, recurringDate)
  - Constraining facets for each datatype (minLength, maxLength, pattern, precision, etc.)
- **The following is based on:**
  - XML Schema Part 0: Primer (2.5.2001)

17

Reminder: DTD declarations

- <!ELEMENT name (fname+, lname)>
- <!ELEMENT address (name, street, (city, state, zipcode) | (zipcode, city))>
- <!ELEMENT contact (address, phone*, email?)>
- <!ELEMENT fname (#PCDATA)>

18

A sample document

```
<?xml version="1.0"?>
<purchaseOrder orderDate=1999-10-20">
  <shipTo country="US">
      <name>Alice Smith</name>
      <street>123 Maple Street</street>
      <city>Mill Valley</city>
      <state>CA</state>
      <zip>90952</zip>
  </shipTo>
```

19

Continues...

```
<billTo country="US">
      <name>Robert Smith</name>
      <street>8 Oak Avenue</street>
      <city>Old Town</city>
      <state>PA</state>
      <zip>95819</zip>
</billTo>
<comment>Hurry, my lawn is going wild!</comment>
```

20

... continues

```
  <items>
    <item partNum="872-AA">
      <productName>Lawnmower</productName>
      <quantity>1</quantity>
      <price>148.95</price>
      <comment>Confirm this is electric</comment>
    </item>
    <item partNum="926-AA">
      <productName>Baby Monitor</productName>
      <quantity>1</quantity>
      <price>39.98</price>
      <shipDate>1999-05-21</shipDate>
    </item>
  </items>
</purchaseOrder>
```

21

DTD

```
<!ELEMENT purchaseOrder  (shipTo, billTo, comment?, items) >
<!ATTLIST purchaseOrder orderDate CDATA #REQUIRED>
<!ELEMENT shipTo (name, street, city, state, zip)>
<!ATTLIST shipTo country NMTOKEN #REQUIRED>
<!ELEMENT billTo (name, street, city, state, zip)>
<!ATTLIST billTo country NMTOKEN #REQUIRED>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT items (item*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT street (#PCDATA)>
```

22

DTD continues

```
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT item (productName, quantity, USPrice,  comment?,
                shipDate?)>
<!ATTLIST item partNum CDATA #REQUIRED>
<!ELEMENT productName (#PCDATA)>
<!ELEMENT quantity (#PCDATA)>
<!ELEMENT USPrice (#PCDATA)>
<!ELEMENT shipDate (#PCDATA)>
```

23

In this part...

- complex vs. simple types
- shared types and element declarations
- occurrence constraints (elements/attributes)
- global vs. local (types/elements/attr.)
- simple types
- special cases for element content

24

Complex and simple types

- Schema defines types for elements and attributes
- complex types: allow elements in their content and may have attributes
- simple types: cannot have element content and cannot have attributes
- elements can have complex or simple types, attributes can have simple types

25

XML Schema: structure

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation> … </xsd:annotation>
<xsd:element name="purchaseOrder" type="PurchaseOrderType"/>
<xsd:element name="comment" type="xsd:string"/>
<xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>… </xsd:sequence>
  <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
…
</xsd:schema>
```

26

USAddress type

```
<xsd:complexType name="USAddress" >
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string" />
    <xsd:element name="street" type="xsd:string" />
    <xsd:element name="city" type="xsd:string" />
    <xsd:element name="state" type="xsd:string" />
    <xsd:element name="zip" type="xsd:decimal" />
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:NMTOKEN"
      fixed="US" />
</xsd:complexType>
```

27

PurchaseOrderType

```
<xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
    <xsd:element name="shipTo" type="USAddress" />
    <xsd:element name="billTo" type="USAddress" />
    <xsd:element ref="comment" minOccurs="0" />
    <xsd:element name="items" type="Items" />
  </xsd:sequence>
  <xsd:attribute name="orderDate" type="xsd:date" />
</xsd:complexType>
```

28

Shared types, references

- element declarations for "shipTo" and "billTo" associate different element names with the same complex type
- attribute declarations must reference simple types
- element "comment" declared on the top level of the schema (here reference only)

29

Occurrence constraints

- minOccurs, maxOccurs (defaults: 1)
  - minOccurs: minimun number of times an element may appear
  - element is optional, if minOccurs = 0
  - maxOccurs: maximum number of times an element may appear
- attributes may appear once or not at all

30

Attributes use, default and fixed (in attribute declarations)

- Attribute "use" is used in an attribute declaration to indicate whether the attribute is 'required', 'optional' or 'prohibited'
- default value may be provided if 'optional' is set
  - if the instance does not give the value the default is used

31

Attributes use, default and fixed (in attribute declarations)

- Attribute "fixed"
  - the value of the attribute is the value of "fixed"

<xsd:attribute name="temp1" type="xsd:decimal" use="optional" default="37" /> (if attribute not present -> value=37)

<xsd:attribute name="temp2" type="xsd:decimal" use="optional" fixed="37" /> (if attribute not present -> value=37)

<xsd:attribute name="temp2" type="xsd:decimal" use="required" fixed="37" /> (attribute has to be present and its value must be 37)

32

Global vs. local

- named types vs. anonymous types
- global elements/attributes vs. local elements/attributes

33

```
<xsd:complexType name="Items">
 <xsd:sequence>
  <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType><xsd:sequence>
      <xsd:element name="productName" type="xsd:string" />
      <xsd:element name="quantity">
        <xsd:simpleType>
          <xsd:restriction base="xsd:positiveInteger">
            <xsd:maxExclusive value="100"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="USprice" type="xsd:decimal"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="shipDate" type="xsd:date"
                   minOccurs="0"/>
    </xsd:sequence>
     <xsd:attribute name="partNum" type="Sku" use="required"/>
   </xsd:complexType>
  </xsd:element></xsd:sequence>
</xsd:complexType>
```

Anonymous type definitions

- Schemas can be constructed by defining sets of **named types** such as PurchaseOrderType on the top level and then declaring elements such as purchaseOrder
- if a type is used only once, it is more compactly defined as an **anonymous type**

35

Anonymous type definitions

- You can define anonymous types by the lack of 'type=' in an element declaration and by the presence of an unnamed (simple or complex) type definition following the element name
  - see the Items type definition

36

Global elements and attributes

- Global elements and attributes have declarations that appear as the children of the schema element
- global elements and attributes can be referenced in one or more declarations using the "ref" attribute

37

Global elements and attributes

- global elements can appear in the instance document in the place where they have been referenced, or at the top level of the document
- global declarations cannot contain references
- global declarations cannot contain occurrence constraints

38

Local elements/attributes

- local elements and attributes are declared within a type definition
- make context-sensitive structures possible (not possible with DTDs)
  - <book><title>Of Mice and Men</title>...</book>
  - <person><title>Mrs.</title>...</person>

39

Simple types

- built-in types, derived types
- facets
- list types, union types

40

Simple types

- Built-in types
  - e.g. string, integer, positiveInteger, decimal, float, boolean, time, date, recurringDay, uriReference, language, ID, IDREF
  - must have XML Schema namespace prefix
- derived types
  - derived from built-in and other derived types
  - by defining restrictions to the base type
    - each base type has a set of facets that can be used for restrictions

41

Facets

- XML Schema defines 15 facets
  - e.g. string has facets: length, minLength, maxLength, pattern, enumeration
  - e.g. integer has facets: pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive, precision

42

Defining a new type of integer

- New type whose range of values is between 10000 and 99999

```
<xsd:simpleType name="myInteger">
  <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="10000"/>
      <xsd:maxInclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

43

Patterns

```
<xsd:simpleType name="Sku">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  <xsd:restriction>
</xsd:simpleType>
```

- "three digits followed by a hyphen followed by two upper-case ASCII letters"

44

Enumeration facet

- Limits values to a set of distinct values

```
<xsd:simpleType name="USState">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="AK"/>
      <xsd:enumeration value="AL"/>
      <xsd:enumeration value="AR"/>
      <!-- and so on -->
   </xsd:restriction>
</xsd:simpleType>
```

45

List types

- List types are comprised of sequences of atomic types

```
<xsd:element name="listOfMyInt" type="listOfMyIntType" />
<xsd:simpleType name="listOfMyIntType">
   <xsd:list itemtype="myInteger"/>
</xsd:simpleType>

instance:
<listOfMyInt>20003 15037 95977 95945</listOfMyInt>
```

46

Union types

- Type can be chosen from a set:

```
<xsd:element name="zips" type="zipUnion"/>
<xsd:simpleType name="zipUnion">
   <xsd:union memberTypes="USState listOfMyIntType"/>
</xsd:simpleType>

<zips>CA</zips>
<zips>95630 95977 95945</zips>
```

47

Special cases for element content

- simple type content + attributes
- mixed content
- empty content
- anyType

48

Simple type content + attributes

- How to define attributes for elements with simple type content?
  - In instance: <internationalPrice currency="EUR">423.45</internationalPrice>
  - in the sample schema: <xsd:element name="USPrice" type="xsd:decimal"/> comes close
- but simple types cannot have attributes
- -> a complex type has to be defined

49

Simple type content + attributes

- New complex type is derived from type 'decimal'

```
<xsd:element name="internationalPrice>
 <xsd:complexType>
  <xsd:simpleContent>
   <xsd:extension base="xsd:decimal">
    <xsd:attribute name="currency" type="xsd:string" />
   </xsd:extension>
  </xsd:simpleContent>
 </xsd:complexType>
</xsd:element>
```

50

Mixed content

- Element contains both character data and subelements

```
<letterBody>
<salutation>Dear Mr.<name>Robert Smith</name>.</salutation>
Your order of <quantity>1</quantity> <productName>Baby
Monitor</productName> shipped from our warehouse on
<shipDate>1999-05-21</shipDate> …
</letterBody>
```

51

Mixed content

```
<xsd:element name="letterBody">
 <xsd:complexType mixed="true">
  <xsd:sequence>
   <xsd:element name="salutation">
    <xsd:complexType mixed="true">
     <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="quantity" type="xsd:positiveInteger"/>
   …
  </xsd:sequence></xsd:complexType></xsd:element>
```

52

Empty content

- Assume we want the **internationalPrice** element to have both the unit of currency and the price as attribute values:
  - **<internationalPrice currency="EUR" value="423.45" />**
- i.e. the element has no content
- solution: no elements defined in the content model

53

Empty content

```
<xsd:element name="internationalPrice"
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:restriction base:"xsd:anyType">
        <xsd:attribute name="currency" type="xsd:string" />
        <xsd:attribute name="value" type="xsd:decimal" />
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

54

Shorthand for empty complex type

```
<xsd:element name="internationalPrice"
  <xsd:complexType>
        <xsd:attribute name="currency" type="xsd:string" />
        <xsd:attribute name="value" type="xsd:decimal" />
  </xsd:complexType>
</xsd:element>
```

55

anyType

- The anyType seen in the definition for an empty content model represents an abstraction which is the base type from which all simple and complex types are derived
- anyType does not constrain its content in any way
- can be used like other types
- is a default if no type is specified
  - <xsd:element name="anything" />

56