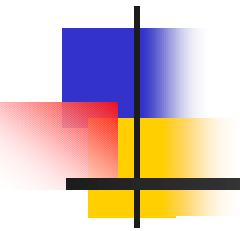


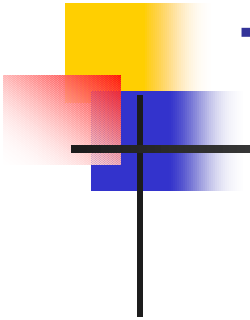
How to develop Syntax and XML Schema

Ted Leung

Chairman, ASF XML PMC

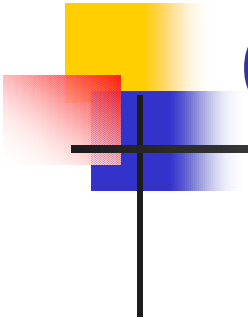
Principal, Sauria Associates, LLC

twl@sauria.com



Thank You

- ASF
- Xerces-J-Dev
- Xerces-J-User



Outline

- Overview
- xml.apache.org project
- Business / Technical motivation for XML Schema
- Coverage of Key XML Schema Features
- Alternative Schema Languages

Apache Software Foundation

- Non-profit foundation
  - volunteers
- Open Source Software
- Apache License
  - Not viral like GPL
  - Commercial Use is fine
- Projects
  - Web Server
  - jakarta.apache.org
    - JSP, Servlets, Ant, Struts, Server Frameworks
  - xml.apache.org

xml.apache.org

- Part of the Apache Software Foundation
- Open Source XML processing components
  - XML Parser [Xerces] (Java, C++, Perl)
  - XSLT processor [Xalan] (Java, C++)
  - XSL Formatting Objects [FOP]
  - SVG [Batik]
  - Cocoon
  - SOAP [Apache-SOAP / Axis]

Business Motivation

- XML is a language of agreements
- We needed a way to specify those agreements in more detail – raises the level of discourse between applications
- We needed a way for machines to be able to do more with those agreements

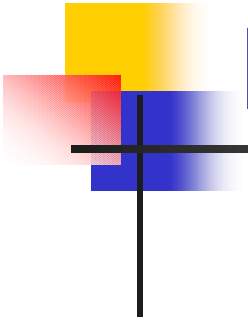
Technical Motivation

- Three major needs leading up to XML Schema
  - Strong data typing of element content and attributes – push more validation into the XML infrastructure
  - Integration of namespaces into grammars
  - Use of XML syntax to describe the grammar

When would you use Schema?

- You have rich data types [typing]
- You need open content models [wildcards]
- You need to combine data from multiple organzations [namespaces]
- You are mapping from database [uniqueness]

Example Schema Applications



- SOAP / WSDL
- XSLT 2
- XForms
- Many more to come

XML Schema Specification

- W3C Recommendation as of 5/2/2001
- Three documents:
  - XML Schema Part 0: Primer
  - XML Schema Part 1: Structures
  - XML Schema Part 2: Datatypes
- Support in Xerces-J 1.4+

Type System Design Features

- Elements and attributes have explicit types
- Types can be defined independently
- Definitions
  - Global
  - Local/Anonymous

Two kinds of types

- Simple types
  - describe character data
- Complex types
  - Can have attributes
  - Can have content models (elements)

Example of simple type

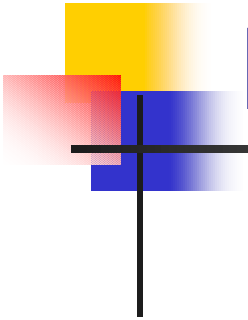
```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
 <element name='withdraw' type="integer“/>
</schema>
```

Simple types

- Lexical Space
- Value Space
- Facets
- Primitive built-in types
  - XML 1.0 types
  - String, boolean, numbers, dates, times

Creating new simple types

- Restriction
  - ```
    <schema
    xmlns='http://www.w3.org/2001/XMLSchema'>
    <element name='withdraw'>
     <simpleType>
      <restriction base='integer'>
       <minInclusive value="0"/>
      </restriction>
     </simpleType>
    </element>
    </schema>
    ```
- Some built in types are restrictions of primitive types
    ```



Facets

- Numeric range restriction
  - Miniumum and maximum
  - Inclusive / exclusive
- Enumeration
  - Explicit enumeration
- Pattern – regular expressions
  - Constrains strings

Atomic vs non-Atomic types

- Atomic Types
- List
  - ```
    <schema
    xmlns='http://www.w3.org/2001/XMLSchema'>
     <element name='friends'>
      <simpleType>
      <list itemType='string'/>
      </simpleType>
     </element>
    </schema>
    ```
- Union
    ```

Example of complex type

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
 <element name='withdraw'>
  <complexType>
   <simpleContent>
      <extension base='integer'>
       <attribute name='currency‘ type='string'
                  default='us'/>
      </extension>
   </simpleContent>
  </complexType>
 </element>
</schema>
```

Complex types

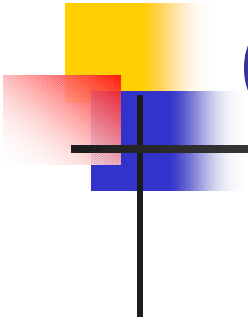
- Carry attributes
- Can have child element content
- Can be derived from simple types

Content Models

- Complex types have content models to describe nested elements, etc.
- Content Model types
  - SimpleContent – just content
  - element only content
  - ComplexContent – type derivation
  - Mixed Content – elements and content
  - Empty content

Element Only Content

```
<schema
    xmlns='http://www.w3.org/2001/XMLSchema'>
 <complexType name='person'>
  <sequence>
    <element name='name' type='string'/>
    <element name='age' type='positiveInteger'>
  </sequence>
 </complexType>
</schema>
```



Compositors

- Three compositors
  - <sequence>
  - <choice>
  - <all>

Mixed Content

- Stricter than XML 1.0 – order and number of child elements counts
- Attribute on complexType or complexContent

Example of Mixed Content

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
 <element name='boilerplate'>
  <complexType mixed='true'>
   <all>
      <element name='heading' type='string'/>
      <element name='version' type='decimal'/>
      <element name='email' type='string'/>
   </all>
  </complexType>
 </element>
</schema>
```

Empty Content

```
<schema
 xmlns='http://www.w3.org/2001/XMLSchema' >
 <complexType name=‘void’>
  <attribute name=‘size’ type=‘integer’/>
 </complexType>
</schema>
```

minOccurs/maxOccurs

- How do I specify how many times an element occurs?

```
<schema
  xmlns='http://www.w3.org/2001/XMLSchema' >
  <complexType name='friend'>
   <element name='lastName' type='string'/>
   <element name='firstName' type='string'/>
  </complexType>
  <element name='friends'>
   <sequence>
    <element name='friend' minOccurs='0'
             maxOccurs='unbounded'/>
   </sequence>
</schema>
```

Attributes

- Use any simple type

```
<attribute name=‘delayed’ type=‘boolean’
           use=‘optional’ default=‘false’/>

<attribute name=‘ranking’
           type=‘positiveInteger’
           use=‘required’/>
```

ComplexType Extension

- Add to end of type

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
   <complexType name='personType'>
    <sequence>
     <element name='name' type='string'/>
     <element name='father' type='string'/>
    </sequence>
   </complexType>
```

ComplexType Extension

```
<complexType name='parentType'>
 <complexContent>
  <extension base='personType'>
     <sequence>
      <element name='child' type='string'/>
      </sequence>
  </extension>
 </complexContent>
</complexType>
<element name='person' type='personType'/>
<element name='parent' type='parentType'/>
</schema>
```

ComplexType Restriction 1

- Must state restriction

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
 <complexType name='personType'>
  <sequence>
   <element name='name' type='string'/>
   <element name='father' type='string'/>
  </sequence>
 </complexType>
```

ComplexType Restriction 2

```
<complexType name='orphanType'>
  <complexContent>
   <restriction base='personType'>
    <sequence>
     <element name='name' type='string'/>
    </sequence>
   </restriction>
  </complexContent>
 </complexType>

 <element name='person' type='personType'/>
 <element name='orphan' type='orphanType'/>
</schema>
```

Modularity constructs

- <group>
  - For Content Models
- <attributeGroup>
  - For attributes

Modularity Example

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
 <group name='phoneNumber'>
   <sequence>
    <element name='areaCode' type='positiveInteger'/>
    <element name='number' type='string'/>
   </sequence>
 </group>
 <attributeGroup name='callingAttributes'>
   <attribute name='callingCard' type='string'/>
 </attributeGroup>
 <element name='ISP'>
   <complexType>
    <group ref='phoneNumber'/>
    <attributeGroup ref='callingAttributes'/>
   </complexType>
 </element>
</schema>
```

Modularity vs inheritance

- Kohsuke Kawaguchi
- Model groups simulate inheritance, since they nest
- Restriction forces you to write it all out, so this is a model group as well
- Checking model groups is much easier than checking inheritance

Include

- For physical modularity of a schema
- Definitions in the same target namespace
- `<include schemaLocation=‘http://www.schemas.com/fragment.xsd’/>`

Target Namespaces

- To put a set of definitions into a namespace...

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'
   targetNamespace='http://www.sauria.com/Schemas/Tutori
   al/target'>
 <element name='withdraw' type="integer“/>
</schema>
```

Import

- Mix types from different namespaces
- Simple types can be used
  - Must be global
- Complex types can be used
  - Named, global types only
- Import element must appear first in schema
- Can also import and redefine imported items

Import Example 1

```
<schema
   xmlns='http://www.w3.org/2001/XMLSchema
   xmlns:beers='http://www.sauria.com/Schemas/Tuto
   rial/importee'
   targetNamespace='http://www.sauria.com/Schemas/
   Tutorial/importee‘
   elementFormDefault=‘qualified’>
   <complexType name='importedBeer'>
    <sequence>
     <element name='color' type='string'/>
     <element name='alcohol' type='decimal'/>
    </sequence>
  </complexType>
</schema>
```

Import Example 2

```
<schema
   xmlns='http://www.w3.org/2001/XMLSchema‘
   xmlns:importedBeerSchema='http://www.sauria.com
   /Schemas/Tutorial/importee‘
   targetNamespace='http://www.sauria.com/Schemas/
   Tutorial/importer‘
   elementFormDefault=‘qualified’>

   <import
   namespace='http://www.sauria.com/Schemas/Tutori
   al/importee‘ schemaLocation='importee.xsd'/>

   <element name='beer‘
          type='importedBeerSchema:importedBeer'/>

   </schema>
```

XML Schema Instance NS

- XML Schema Instance Namespace
  - URI is http://www.w3.org/2001/XMLSchema-instance
  - Prefix is xsi
- type
  - Force an element to be associated with a particular type
  - `<person xsi:type='parent'/>`
- nil
  - For values that can be nil/null, specify the nil value.

xsi:nil Example

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
 <complexType name='personType'>
  <sequence>
   <element name='name' type='string'/>
   <element name='father' type='string'/>
  </sequence>
 </complexType>
 <element name='person' type='personType'
  nillable='true'/>
</schema>


<person
  xmlns:xsi=‘http://www.w3.org/2001/XMLSchema-
  instance’
  xsi:noNamespaceSchemaLocation='nil.xsd'
  xsi:nil='true'/>
```

Wildcards

- How do I leave a schema open for extension?
- “Any element from namespace x,y or z”
- “Any element from a namespace besides this one”
- “Any element from a schema in no namespace”
- “Any element from any namespace”
- Similarly for attributes

Wildcard Example 1

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'
   xmlns:ann='http://www.sauria.com/Schemas/Tutori
   al/annonate'
   targetNamespace='http://www.sauria.com/Schemas/
   Tutorial/annonate‘
   elementFormDefault=‘qualified’>

   <complexType name='annotated'>
    <sequence>
      <element name='partName' type='string'/>
      <element name='partNo' type='integer'/>
     <element name='annotation'
       type='ann:annotation'/>
    </sequence>
  </complexType>
```

Wildcard Example 2

```
 <complexType name='annotation'>
  <sequence>
   <any namespace='http://www.w3.org/1999/xhtml'
  processContents='lax'/>
  </sequence>
 </complexType>

 <element name='part' type='ann:annotated'/>
</schema>
```

Annotation

- A standard way to document schemas
- `<documentation>` – for humans
- `<appinfo>` – for programs
-
```
<annotation>
 <documentation>text here
 </documentation>
 <appinfo>
  <rdf-loc>http://www.schemas.com/rdf
  </rdf-loc>
 </appinfo>
</annotation>
```

Uniqueness

- “I want to ensure that the value of an element <foo> is unique”
  - In what scope? There may be two elements named <foo>, each children of two different elements
  - Specify the scope using XPath

Unique Example 1

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
 <complexType name='bookType'>
   <sequence>
    <element name='title' type='string'/>
    <element name='ISBN' type='string'/>
   </sequence>
 </complexType>

 <element name='books'>
   <complexType>
    <sequence>
     <element name='book' type='bookType'
   maxOccurs='unbounded'/>
    </sequence>
   </complexType>
 </element>
```

Unique Example 2

```
 <unique name='isbn'>
  <selector xpath='books/book'/>
  <field xpath='ISBN'/>
 </unique>
</schema>
```

Keys

```
<schema xmlns='http://www.w3.org/2001/XMLSchema'>
 <complexType name='bookType'>
  <sequence>
   <element name='title' type='string'/>
   <element name='ISBN' type='string'/>
  </sequence>
 </complexType>

 <complexType name='cardCatalog'>
  <sequence>
   <element name='cardNumber' type='string'/>
  </sequence>
 </complexType>
```

Keys Example 1

```
<element name='books'>
 <complexType>
  <sequence>
     <element name='book' type='bookType'
 maxOccurs='unbounded'/>
     <element name='catalogCard'
 type='cardCatalog' maxOccurs='unbounded'/>
  </sequence>
 </complexType>
</element>
```

Keys Example 2

```
<unique name='isbn'>
   <selector xpath='books/book'/>
   <field xpath='ISBN'/>
 </unique>

 <key name='card'>
   <selector xpath='books/book'/>
    <field xpath='ISBN'/>
 </key>

 <keyref name='cardRef‘ refer=‘card’>
   <selector xpath='books/catalogCard'/>
   <field xpath='cardNumber'/>
 </keyref>
</schema>
```

Referencing a schema

- ```
  <?xml version="1.0" encoding="UTF-8"?>
  <withdraw
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:noNamespaceSchemaLocation='simple.xsd'>
  25
  </withdraw>
  ```

- ```
  <?xml version="1.0" encoding="UTF-8"?>
  <bank:withdraw
  xmlns='http://www.schemas.com/bank'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:schemaLocation='http://www.schemas.com/bank
  simple.xsd'>
  25
  </bank:withdraw>
  ```

Tool support

- Xerces-J [REC]
- XSV [REC]
- Oracle XML Parser [PR]
- MSXML 4.0 [PR]

Cool Tool Tricks

- Use XSLT on your Schema
- Data Binding

Alternatives to XML Schema

- Some people feel XML Schema is too complicated
- Relax => TREX => Relax NG
  - Relax (Murata Makoto)
    - Relax Verifier
  - TREX (James Clark)
    - JTREX
  - Relax NG (OASIS)
    - Jing

Relax NG

- Uses XML Instance Syntax
- Supports namespaces
- Much simpler than XML Schema
- More orthogonal than XML Schema
- Can use XML Schema datatypes
- Doesn’t support uniqueness, inheritance

Relax NG Example

```
<element name='book' dataTypeLibrary='…'>
 <element name='title'>
   <data type='string'>
 </element>
 <element name='quantity'>
   <data type='integer'>
</element>
</element>

<element name='books'>
 <oneOrMore>
   <ref name='book'/>
 </oneOrMore>
</element>
```

Thank You!

- http://xml.apache.org
- twl@apache.org