
26 November 2007

David J. Malan
malan@post.harvard.edu
Last Time

XQuery 1.0 and DTD

- XQuery 1.0
- DTD
- Project 3


## Last Time

### XQuery 1.0 and DTD

```xml
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE BookStore [  
<!ELEMENT BookStore (Book*)>  
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>  
<!ELEMENT Title (#PCDATA)>  
]>
<BookStore>
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>1998</Date>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
  <Book>
    <Title>Illusions The Adventures of a Reluctant Messiah</Title>
    <Author>Richard Bach</Author>
    <Date>1977</Date>
    <Publisher>Dell Publishing Co.</Publisher>
  </Book>
...  
</BookStore>
```

Computer Science E-259

This Time

- XML Schema (Second Edition)
- Project 4
XML Schema (Second Edition)

History

- After the release of XML 1.0, DTDs were soon recognized as insufficient
- Work towards new schema standards began in early 1998
- Different companies all proposed different variations of schema formats defined in XML; all submitted as Notes to the W3C
  - XML Data (MS, Arbortext, Inso), January 1998
  - DCD (MS & IBM), June 1998
  - XDR (XML Data Reduced), July 1998
  - SOX (Schema for OO XML), July 1999
History

- W3C Working Group formed to address the schema issue in early 1999
- XML Schema became an official recommendation in May 2001; Second Edition in October 2004
  - Primer
  - Structures
  - DataTypes
Let's look at *po.xml* and *po.xsd*

Notice that

- XML instance points to schema
- XML Schema declares elements
- XML Schema defines types
- Types come in a number of varieties
  - Built-in types (*e.g.*, `xsd:string`, `xsd:date`)
  - Simple types
  - Complex types
XML Schema (Second Edition)

Why?

- Data validation
  - Structure of elements and attributes
  - Order of elements
  - Data values of elements and attributes
  - Uniqueness of values
- Establish a contract with trading partners
- Documentation
- Augmentation of instance with default values
- Storage of application information
Another Example

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="product" type="ProductType"/>
  <xsd:complexType name="ProductType">
    <xsd:sequence>
      <xsd:element name="number" type="xsd:integer"/>
      <xsd:element name="size" type="SizeType"/>
    </xsd:sequence>
    <xsd:attribute name="effDate" type="xsd:date"/>
  </xsd:complexType>
  <xsd:simpleType name="SizeType">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="2"/>
      <xsd:maxInclusive value="18"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>

<product effDate="2001-04-02">
  <number>557</number>
  <size>10</size>
</product>
```
Declarations v. Definitions

- Declarations used for components that can appear in the instance (e.g., elements and attributes)
- Definitions used for components internal to the schema (e.g., data types and model groups)
- Order in schema document is insignificant
Global v. Local Components

- Global components
  - Appear at the top level of the schema (children of `<xsd:schema>`)
  - Name must be unique in component type in schema
- Local components
  - Scoped to the definition or declaration that contains them
  - For example, elements declared in the scope of a complex type or types declared anonymously inside other constructs
XML Schema (Second Edition)

Element and Attribute Declarations

- The basic building blocks of XML documents
- Each associated with a data type
  - Use different names for data that is structurally the same by sharing a type (e.g., shipTo and billTo both have type USAddress)
  - Use the same names but two different types in different contexts (e.g., size child of shirt with type xsd:integer or size child of hat with enumerated type "S", "M", "L")
Simple v. Complex Types

- Elements with simple types have character data content but no child elements or attributes
  
  `<size>10</size>`
  `<comment>Extra trim on sides</comment>`
  `<availableSizes>10 large 2</availableSizes>`

- Elements with complex types can have child elements or attributes
  
  `<size system="US-DRESS">10</size>`
  `<comment>Extra <b>trim</b> on sides</comment>`
  `<availableSizes>`
    `<size>10</size>`
    `<size>2</size>`
  `</availableSizes>`

- Attributes always have simple types
Named v. Anonymous Types

- Named types are always defined globally and are available for reuse.
- Anonymous types have no names and are local to an element or attribute declaration.

```xml
<xsd:element name="size">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="2"/>
      <xsd:maxInclusive value="18"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```
Type Definition Hierarchy

- Data types can be derived from other types by restricting or extending
- In our example, SizeType restricts the range of an integer
- Complex type UKAddressType can extend AddressType by adding more children
- Most importantly, a subtype can be substituted when a base type is expected (a UKAddressType is valid when AddressType is expected)
Simple Types

- Built-In Data Types
  - string-related
    - ENTITIES, ENTITY, ID, IDREF, IDREFS, language, Name, NCName, NMTOKEN, NMTOKENS, normalizedString, QName, string, token
  - Date-related
    - date, dateTime, duration, gDay, gMonth, gMonthDay, gYear, gYearMonth, time
  - Number-related
    - base64Binary, byte, decimal, double, float, hexBinary, int, integer, long, negativeInteger, nonPositiveInteger, positiveInteger, short, unsignedLong, unsignedInt, unsignedShort, unsignedByte
  - Err, unrelated
    - anyURI, boolean, NOTATION, ...

- New simple types can be derived from built-in ones by restricting them along some facets (e.g., minInclusive)

- Most simple types are atomic types but we can also have:
  - List types: whitespace-separated lists of atomic values
  - Union types: have a value picked from a set of types
Complex Types

- Contents of an element are character data and child elements
- Four different content types:
  - Simple, Element, Mixed, Empty
- Content Models describe the order and structure of child elements of a complex type
  - sequence groups specify order
  - choice groups allow one of several options
  - all groups require all child elements appear 0 or 1 times in any order
Another Example

```xml
<xsd:complexType name="ProductType">
  <xsd:sequence>
    <xsd:element name="number" type="xsd:integer"/>
    <xsd:choice minOccurs="0" maxOccurs="3">
      <xsd:element name="size" type="SizeType"/>
      <xsd:element name="color" type="ColorType"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:any/>
  <xsd:attribute name="effDate" type="xsd:date"/>
</xsd:complexType>
```
Namespaces

- Namespaces are used heavily in XML Schema, so let's review.
- A namespace is bound to a URI such as http://example.org/prod or urn:example:org.
- An instance can include one or more namespaces by mapping element prefixes to namespace URIs.

```xml
<prod:product xmlns:prod="http://example.org/prod">
    <prod:number>557</prod:number>
    <prod:size>10</prod:size>
</prod:product>
```

- Prefix choice doesn't matter; only the mapping to the URI does (conventions exist like xsl:, xsd:).
Multiple Namespaces

- Multiple namespace declarations are easy and useful

```xml
<ord:order xmlns:ord="http://example.org/ord"
            xmlns:prod="http://example.org/prod">
  <ord:number>123ABBCC123</ord:number>
  <ord:items>
    <prod:product>
      <prod:number>557</prod:number>
      <prod:size system="US-DRESS">10</prod:size>
    </prod:product>
  </ord:items>
</ord:order>
```

- Note that `number` appears twice in two different namespaces
XML Schema (Second Edition)

Default Namespaces

- A default namespace declaration binds elements with no prefix to a namespace

```xml
<order xmlns="http://example.org/ord"
       xmlns:prod="http://example.org/prod">
  <number>123ABBC123</number>
  <items>
    <prod:product>
      <prod:number>557</prod:number>
      <prod:size system="US-DRESS">10</prod:size>
    </prod:product>
  </items>
</order>
```
Target Namespaces

- XML Schema lets you specify at most one namespace as the target namespace
- All declarations and definitions will be part of the target namespace

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://example.org/prod"
            targetNamespace="http://example.org/prod">
  <xsd:element name="product" type="ProductType"/>
  <xsd:element name="number" type="xsd:integer"/>
  <xsd:complexType name="ProductType">
    <xsd:sequence>
      <xsd:element ref="number"/>
      <xsd:element ref="size"/>
    </xsd:sequence>
  </xsd:complexType>
  ...
</xsd:schema>
```
Relating Instances to Schemas

- There are four ways to relate instances to schemas
  - Use a hint in the instance (\texttt{xsi:schemaLocation} or \texttt{xsi:noNamespaceSchemaLocation} on root element points to schema)
  - Let the application choose and pass to schema validator or parser using code
  - Let the user choose (a dialog for example)
  - Dereference the namespace URI to locate a schema
Another Example

```xml
<prod:product xmlns:prod="http://example.org/prod"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://example.org/prod prod.xsd">
  <prod:number>557</prod:number>
  <prod:size>10</prod:size>
</prod:product>

<order xmlns="http://example.org/ord"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://example.org/prod prod.xsd
                          http://example.org/ord ord.xsd">
  <number>123ABBCC123</number>
  <items>
    <product xmlns="http://example.org/prod">
      <number>557</number>
      <size system="US-DRESS">10</size>
    </product>
  </items>
</order>
```
XML Schema (Second Edition)

Schema Processors

- XSV (XML Schema Validator)
  - Not a parser but just a schema validator
  - http://www.w3.org/2001/03/webdata/xsv
- Xerces
  - Turn on parser feature asking for validation
- Stylus Studio
- XMLSpy
- ...
<BookStore xmlns="http://www.books.org"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.books.org BookStore.xsd">
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>1998</Date>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
    <Book>
        <Title>Illusions The Adventures of a Reluctant Messiah</Title>
        <Author>Richard Bach</Author>
        <Date>1977</Date>
        <Publisher>Dell Publishing Co.</Publisher>
    </Book>
    ...
</BookStore>
Project 4

Scamazon.com

catalog.xml + catalog.xsl
Cart + cart.xsl
PO-ACK + confirm.xsl

Catalog servlet
Cart servlet

PO
PO-ACK

Purchasing web service

Warehouse
Next Time

XML Schema 1.0, Continued

- Datatypes
- Structures
- Simple Types
- Complex Types

26 November 2007

David J. Malan
malan@post.harvard.edu