Computer Science E-259

XML with J2EE

Lecture 9: XML Schema (Second Edition)

6 April 2006

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 version="1.0" standalone="yes"?>
<!DOCTYPE BookStore [


]>
<BookStore>
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>1998</Date>
    <ISBN>1-56592-235-2</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
  <Book>
    <Title>Illusions The Adventures of a Reluctant Messiah</Title>
    <Author>Richard Bach</Author>
    <Date>1977</Date>
    <ISBN>0-440-34319-4</ISBN>
    <Publisher>Dell Publishing Co.</Publisher>
  </Book>
  ...
</BookStore>
```

Adapted from http://www.xfront.com/xml-schema.html.

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

XML Schema (Second Edition)

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

XML Schema (Second Edition)

By Example

- 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

XML Schema (Second Edition)

Another Example

```
<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>
```

XML Schema (Second Edition)

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

XML Schema (Second Edition)

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 but two different types in different contexts (*e.g.,* **`size`** child of **`shirt`** with type **`xsd:integer`** or child of **`hat`** with enumerated type **`"S"`**, **`"M"`**, **`"L"`**)

XML Schema (Second Edition)

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

XML Schema (Second Edition)

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

```
<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>
```

XML Schema (Second Edition)

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)

XML Schema (Second Edition)

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

XML Schema (Second Edition)

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

XML Schema (Second Edition)

Another Example

```
<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:any/>
  </xsd:sequence>
  <xsd:attribute name="effDate" type="xsd:date"/>
</xsd:complexType>
```

XML Schema (Second Edition)

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 namespace by mapping element prefixes to namespace URIs

```
<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:`**)

XML Schema (Second Edition)

Mupltiple Namespaces

- Multiple namespace declarations are easy and useful

```
<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

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

XML Schema (Second Edition)

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

```
<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>
```

XML Schema (Second Edition)

Relating Instances to Schemas

- There are four ways to relate instances to schemas
  - Use a hint in the instance (**`xsi:schemaLocation`** or **`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

XML Schema (Second Edition)

Another Example

```
<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 2.8.0
  - Turn on parser feature asking for validation
- Stylus Studio 2006 XML Enterprise Edition
  - Associate schema with document and click "Validate Document"
- XMLSpy 2006 Enterprise Edition
- ...

XML Schema (Second Edition)

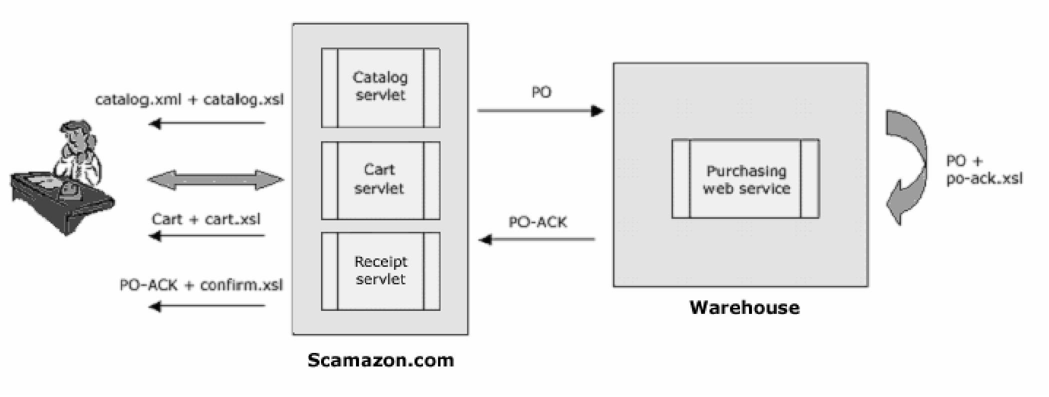
The BookStore, Revisited

```
<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>
    <ISBN>1-56592-235-2</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
  <Book>
    <Title>Illusions The Adventures of a Reluctant Messiah</Title>
    <Author>Richard Bach</Author>
    <Date>1977</Date>
    <ISBN>0-440-34319-4</ISBN>
    <Publisher>Dell Publishing Co.</Publisher>
  </Book>
  ...
</BookStore>
```

Adapted from http://www.xfront.com/xml-schema.html.

Project 4

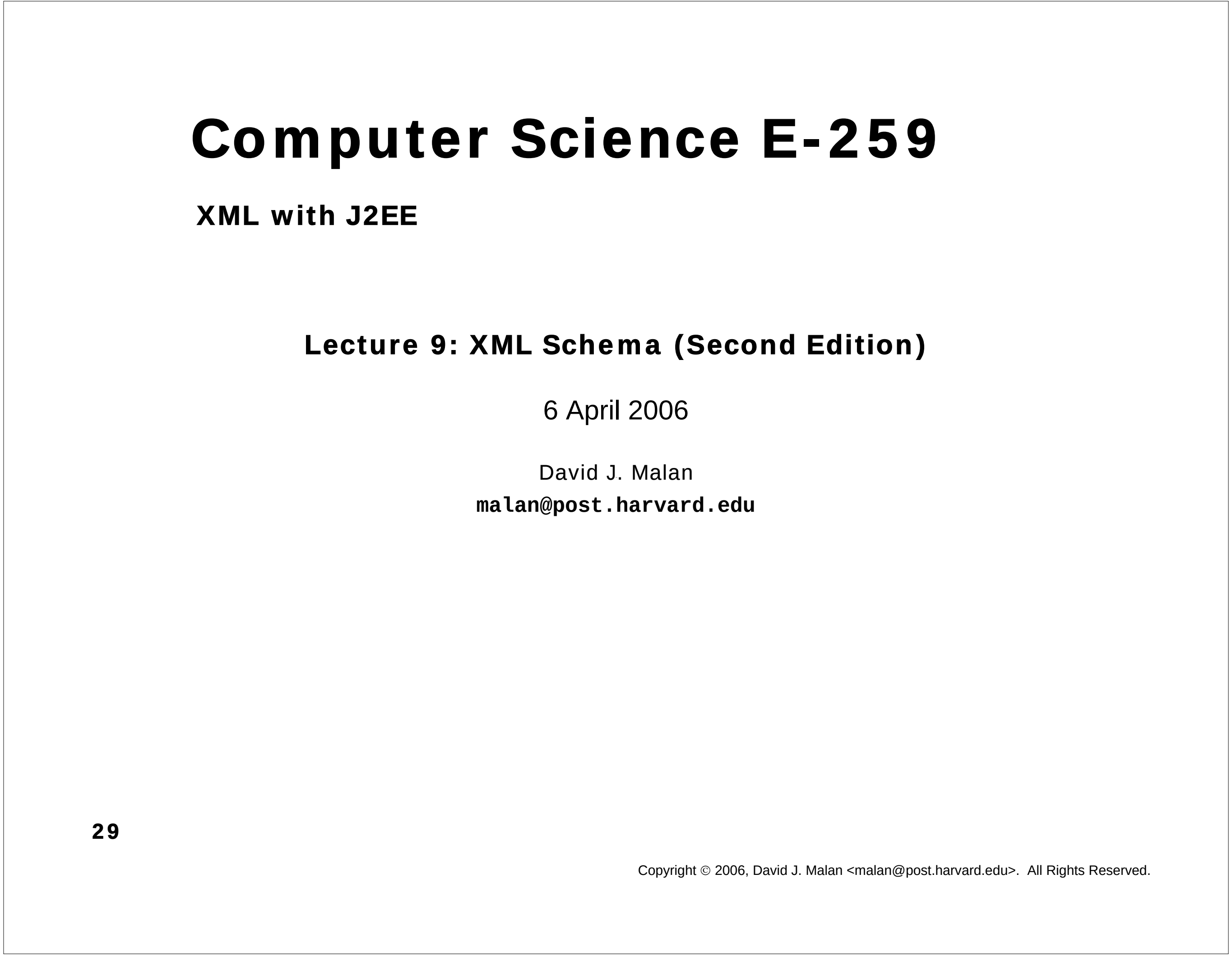
scamazon.com



Next Time

XML Schema 1.0, Continued

- Datatypes
- Structures
- Simple Types
- Complex Types



Computer Science E-259

XML with J2EE

Lecture 9: XML Schema (Second Edition)

6 April 2006

David J. Malan
malan@post.harvard.edu