

XML Schema

- XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents. [www.w3.org/XML/Schema]
- DTD on hormones
- XML Schema is a XML application
 - Follows XML syntax

1

XML Schema Definition (XSD)

- Document model that conforms to the XML Schema standard
 - Translation: XML application with which you can describe other XML applications (document types)
- Set of rules to declare elements, attributes and data types
- <!DOCTYPE schema
PUBLIC "-//W3C//DTD XMLSCHEMA 2001//EN"
SYSTEM http://www.w3.org/2001/XMLSchema.dtd>

2

<schema>

- Root element for document definition
- ```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:csc309="http://www.cs.toronto.edu/csc309"
 targetNamespace="http://www.cs.toronto.edu/csc309" >
<xsd:schema>
```
- Document may include elements and attributes from different schemas (document models)
  - Namespaces are used to disambiguate
  - Defined with `xmlns` attribute
    - `xmlns:prefix=URL`
    - URL uniquely identifies namespace (does not have further meaning)
    - If no prefix is specified, then declaration is for default name space
  - Target namespace
    - Namespace to which the elements, attributes and data types being declared are assigned

5

## <schema>

```
<xsd:schema>
 <!-- Schema Management -->
 <xsd:import> </xsd:import>
 <xsd:include> </>
 <xsd:redefine> </xsd:redefine>
 Used when definitions spread across multiple documents

 <!-- Model-Building Components -->
 <xsd:element> ... </xsd:element>
 <xsd:attribute> ... </xsd:attribute>
 <xsd:simpleType> ... </xsd:simpleType>
 <xsd:complexType> ... </xsd:complexType>

</xsd:schema>
```

4

## <element>

- Creates a new tag
 

```
<xsd:element name="Book" />
```
  - Can limit content to specific type
 

```
<xsd:element name="score" type="integer" default="100" />
<score />
<score>124</score>
```
- Built-in types: anyType, boolean, integer, float, double, decimal, string, nonEmptyString, time, date .....

5

## <element>

- Elements with children
- Option 1
 

```
<xsd:element name="Email">

 <xsd:element name="From" ... </xsd:element>

</xsd:element>
```
- Option 2: References
 

```
<xsd:element name="From" ... </xsd:element>
<xsd:element name="Email">

 <xsd:element ref="From" />

</xsd:element>

<xsd:element name="Folder" <folder>
 </folder>
 <xsd:element ref="Folder" /> <folder> ... </folder>
 </folder>
</xsd:element> </folder>
```

6

## <element>

- Type of elements with children is `complexType`

```
<xsd:element name="Email">
 <complexType>

 </complexType>
</xsd:element>
```

- Content can be one of:

- Empty → 

```
<xsd:element name="PageBreak">
 <complexType />
</xsd:element>
<PageBreak />
```
- `<xsd:sequence>`
- `<xsd:choice>`
- `<xsd:all>`
  - Similar to sequence but in no specific order
- `<xsd:attribute>`

7

## complexType Example

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:csc309="http://www.cs.toronto.edu/csc309"
 targetNamespace="http://www.cs.toronto.edu/csc309" >
```

```
<element name="From" type="string"/>
<element name="To" type="string"/>
<element name="MailingList" type="string"/>
```

```
<element name="Email">
 <complexType>
 <sequence>
 <element ref="csc309:From"/>
 <choice>
 <element ref="csc309:To" />
 <element ref="csc309:MailingList" />
 </choice>
 </sequence>
 </complexType>
</element>
```

8

## Repeatable Elements

- Use attributes `minOccurs` and `maxOccurs`

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:E="http://www.example.org"
 targetNamespace="http://www.cs.toronto.edu/csc309" >

 <xsd:element name="Book"> ... </xsd:element>
 <xsd:element name="Library">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element ref="E:Book" minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

9

## <attribute>

- Creates new attributes
- Empty elements

```
<element name="Image">
 <complexType>
 <!-- No elements – Empty element -->
 <attribute name="Format" type="string" use="optional" />
 <attribute name="Ref" type="string" use="required" />
 </complexType>
</element>
```

```
<Image Format="GIF" Ref="images/boat.gif" />
```

10

## <attribute>

- Elements with content

### Example 1:

```
<element name="Para">
 <complexType mixed="true">
 <attribute name="Language" type="string" />
 </complexType>
</element>
```

```
<Para Language="English">This is an English Paragraph.</Para>
```

### Example 2:

```
<element ... >
 <complexType>
 <sequence>

 </sequence>
 <attribute />
 </complexType>
</element>
```

11

## Constraining Element Content and Attribute Value

- By built-in data type

```
<element name="score" type="integer" default="100" />

<score />
<score>124</score>
```

Built-in types: anyType, boolean, integer, float, double, decimal, string, nonEmptyString, time, date, ID, IDREF, NMTOKEN

12

## Constraining Element Content and Attribute Value

- New simpleType
    - References an existing type through base attribute
    - Restricts or extends existing type
  - Example 1:
 

```
<attribute name="Security">
 <simpleType>
 <restriction base="NMTOKEN">
 <enumeration value="normal" />
 <enumeration value="secret" />
 <enumeration value="topsecret" />
 </restriction>
 </simpleType>
</attribute>
```
  - Example 2
 

```
<simpleType name="internalEmailAddress">
 <restriction base="anyURI">
 <pattern value="[a-zA-Z]+,[a-zA-Z]+@cs.toronto.edu" />
 </restriction>
</simpleType>

<element name="studentEmail" type="internalEmailAddress" />
```
- XML Schema regular expressions reference:  
<http://www.xmlschemareference.com/regularExpression.html>

13

## Constraining Element Content and Attribute Value

- Example 3:
 

```
<element name="stockPrice">
 <simpleType>
 <restriction base="decimal">
 <totalDigits value="5" />
 <fractionDigits value="2" />
 <minInclusive value="0" />
 </restriction>
 </simpleType>
</attribute>
```
- Example 4: Extension based on Unions
 

```
<simpleType name="timeORdate">
 <union memberTypes="time date" />
</simpleType>
```

14

## Named Complex Types

- A complexType that is named in the schema can be set as the type of an element
  - Option 1
 

```
<element name="Email">
 <complexType>
 <sequence>
 <element ref="E:From" />

 </sequence>
 </complexType>
</element>
```
  - Option 2
 

```
<complexType name="eMailType">
 <sequence>
 <element ref="E:From" />

 </sequence>
</complexType>

<element name="Email" type="E:eMailType">
```

15

## Schema Processor

- Validates a document instance against a schema definition
- Similar to an XML Validator
  - XML Document against DTD
- Example
  - Apache Xerces Java parser
  - <http://xml.apache.org/xerces-j/schema.html>
  - java Validator -v library.xml

16