Distributed Computing

Lesson 17: XML Schema

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://www.it-weise.de

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

website

Overview

- How can we specify our own XML formats?

XML Schema

- XML-Schema (W3C, May 2001 [1–6])

XML Schema

- XML-Schema (W3C, May 2001 [1–6])
- Focused on data types

XML Schema

- XML-Schema (W3C, May 2001 [1–6])
- Focused on data types
- Also called: "XML Schema Definition" (XSD)

XML Schema

- XML-Schema (W3C, May 2001 [1–6])
- Focused on data types
- Also called: “XML Schema Definition” (XSD)
- More powerful than DTD (which we do not discuss here)

XML Schema

- XML-Schema (W3C, May 2001 [1–6])
- Focused on data types
- Also called: “XML Schema Definition” (XSD)
- More powerful than DTD (which we do not discuss here)
- XML Schemas are specified in XML $\Longrightarrow$ no new syntax

XML Schema

- XML-Schema (W3C, May 2001 [1–6])
- Focused on data types
- Also called: "XML Schema Definition" (XSD)
- More powerful than DTD (which we do not discuss here)
- XML Schemas are specified in XML $\Longrightarrow$ no new syntax
- Data types: standard types pre-defined, own types can be added

XML Schema

- XML-Schema (W3C, May 2001 [1–6])
- Focused on data types
- Also called: "XML Schema Definition" (XSD)
- More powerful than DTD (which we do not discuss here)
- XML Schemas are specified in XML $\Longrightarrow$ no new syntax
- Data types: standard types pre-defined, own types can be added
- Inheritance

XML Schema

- XML-Schema (W3C, May 2001 [1–6])
- Focused on data types
- Also called: "XML Schema Definition" (XSD)
- More powerful than DTD (which we do not discuss here)
- XML Schemas are specified in XML $\Longrightarrow$ no new syntax
- Data types: standard types pre-defined, own types can be added
- Inheritance
- Support for XML namespaces

XML Schema

- XML-Schema (W3C, May 2001 [1–6])
- Focused on data types
- Also called: “XML Schema Definition” (XSD)
- More powerful than DTD (which we do not discuss here)
- XML Schemas are specified in XML $\Longrightarrow$ no new syntax
- Data types: standard types pre-defined, own types can be added
- Inheritance
- Support for XML namespaces
- Extensible, supports modularization and re-use

Concept

- A document type is defined by

Concept

- A document type is defined by
  - Elements: Where do they appear? How often? Are they optional or required?

Concept

- A document type is defined by
  - Elements: Where do they appear? How often? Are they optional or required?
  - Attributes: Which element has which attributes?

Concept

- A document type is defined by
  - Elements: Where do they appear? How often? Are they optional or required?
  - Attributes: Which element has which attributes?
  - Values: Which values can appear inside elements (between opening and closing tag)? Which values can attributes take on? Numbers? Strings?

Example: Course XML

Listing: An example XML file

```xml
<?xml version="1.0" encoding="UTF-8"?>

<course courseName="Distributed␣Computing">

  <units>60</units>

  <teachers>
    <teacher familyName="Weise" personalName="Thomas" />
    <teacher familyName="Chen"  personalName="Xianglan" />
  </teachers>

  <students>
    <student studentid="SA11111111" score="85.5" />
    <student studentid="SA22222222" score="73.0" />
    <student studentid="SA33333333" score="90.0" />
  </students>

</course>
```

Example: Course XML / element

- We want to define a document schema that allows for exactly one "course" element per document.

Listing: A part of the course XML file

```
<?xml version="1.0" encoding="UTF-8"?>

<course courseName="Distributed␣Computing">
</course>
```

Example: Course XSD / schema & element

- We want to define a document schema that allows for exactly one "course" element per document.
- We define a new *schema* and a namespace for it: `ustc:courses`

Listing: The corresponding part of the courses.xsd file

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs           = "http://www.w3.org/2001/XMLSchema"
           targetNamespace    = "ustc:courses"
           xmlns              = "ustc:courses"
           elementFormDefault = "qualified">

<xs:element name="course">
</xs:element>

</xs:schema>
```

Example: Course XSD / schema & element

- We want to define a document schema that allows for exactly one "course" element per document.
- We define a new *schema* and a namespace for it: `ustc:courses`
- in this schema, we place one `element` tag with the desired name `course`

Listing: The corresponding part of the courses.xsd file

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs                = "http://www.w3.org/2001/XMLSchema"
           targetNamespace     = "ustc:courses"
           xmlns               = "ustc:courses"
           elementFormDefault = "qualified">

<xs:element name="course">
</xs:element>

</xs:schema>
```

Example: Course XML / attributes

- Now we want to add the attribute `courseName`

Listing: A part of the course XML file

```xml
<?xml version="1.0" encoding="UTF-8"?>

<course courseName="Distributed␣Computing">
</course>
```

Example: Course XSD / attributes

- Now we want to add the attribute `courseName`
- Our schema element `course` now becomes *complex*, as it has at least one child attribute or element

Listing: The corresponding part of the courses.xsd file

```
<xs:element name="course">
  <xs:complexType>
    <xs:attribute name="courseName" type="xs:string" />
  </xs:complexType>
</xs:element>
```

Example: Course XSD / attributes

- Now we want to add the attribute `courseName`
- Our schema element `course` now becomes *complex*, as it has at least one child attribute or element
- attributes are declared with `attribute` tag

Listing: The corresponding part of the courses.xsd file

```
<xs:element name="course">
  <xs:complexType>
    <xs:attribute name="courseName" type="xs:string" />
  </xs:complexType>
</xs:element>
```

Example: Course XSD / attributes

- Now we want to add the attribute `courseName`
- Our schema element `course` now becomes *complex*, as it has at least one child attribute or element
- attributes are declared with `attribute` tag
- attributes have types: XSD has lots of pre-defined types

Listing: The corresponding part of the courses.xsd file

```
<xs:element name="course">
  <xs:complexType>
    <xs:attribute name="courseName" type="xs:string" />
  </xs:complexType>
</xs:element>
```

Example: Course XML / nested elements

- Now we want to add an element `units` which must contain one integer number

Listing: A part of the course XML file

```
<course courseName="Distributed␣Computing">

  <units>60</units>
</course>
```

Example: Course XSD / nested elements

- Now we want to add an element `units` which must contain one integer number
- Our *complex* schema element `course` now has at least one child element

Listing: The corresponding part of the courses.xsd file

```
<xs:element name="course">
  <xs:complexType>
    <xs:sequence>

      <xs:element name="units" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example: Course XSD / nested elements

- Now we want to add an element `units` which must contain one integer number
- Our *complex* schema element `course` now has at least one child element
- such nested elements can appear in a `sequence`

Listing: The corresponding part of the courses.xsd file

```
<xs:element name="course">
  <xs:complexType>
    <xs:sequence>

      <xs:element name="units" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example: Course XML / nested elements 2

- Now we want to add the sub-element `teachers` that, in turn, can contain *multiple* `teacher` elements

Listing: A part of the course XML file

```
<course courseName="Distributed␣Computing">
  <teachers>
    <teacher familyName="Weise" personalName="Thomas" />
    <teacher familyName="Chen"  personalName="Xianglan" />
</course>
```

Example: Course XSD / nested elements 2

- Now we want to add the sub-element `teachers` that, in turn, can contain *multiple* `teacher` elements
- `teachers` itself is a complex element, containing a sequence of `teacher` elements

Listing: The corresponding part of the courses.xsd file

```
<xs:element name="teachers">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="teacher" minOccurs="1"
          maxOccurs="unbounded">
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example: Course XSD / nested elements 2

- Now we want to add the sub-element `teachers` that, in turn, can contain *multiple* `teacher` elements
- `teachers` itself is a complex element, containing a sequence of `teacher` elements
- `teacher` can occur at least once (`minOccurs="1"`) and arbitrarily often (`maxOccurs="unbounded"`)

Listing: The corresponding part of the courses.xsd file

```
<xs:element name="teachers">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="teacher" minOccurs="1"
          maxOccurs="unbounded">
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Example: Course XML / attributes 2

- The element `teacher` has two attributes…

Listing: A part of the course XML file

```
<course courseName="Distributed␣Computing">
  <teachers>
    <teacher familyName="Weise" personalName="Thomas" />
    <teacher familyName="Chen"  personalName="Xianglan" />
</course>
```

Example: Course XSD / attributes 2

- The element `teacher` has two attributes. . .
- `teacher` this is also a complex element with two string attributes

Listing: The corresponding part of the courses.xsd file

```
<xs:element name="teacher" minOccurs="1"
    maxOccurs="unbounded">
  <xs:complexType>
    <xs:attribute name="familyName"   type="xs:string"/>
    <xs:attribute name="personalName" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Example: Course XML / elements 3

- After the element `teachers`, `course` contains the element `students` which, in turn, holds a set of `student` elements, each with two attributes (`studentid` and `score`)

Listing: A part of the course XML file

```
<course courseName="Distributed␣Computing">
  <students>
    <student studentid="SA11111111" score="85.5" />
    <student studentid="SA22222222" score="73.0" />
    <student studentid="SA33333333" score="90.0" />
  </students>
</course>
```

Example: Course XSD / elements 3

- After the element `teachers`, `course` contains the element `students` which, in turn, holds a set of `student` elements, each with two attributes (`studentid` and `score`)

Listing: The corresponding part of the courses.xsd file

```
<xs:element name="students">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="student" minOccurs="1"
          maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="studentid" type="xs:string"/>
          <xs:attribute name="score"     type="xs:float"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Listing: The full example courses.xsd file

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs           = "http://www.w3.org/2001/XMLSchema"
           targetNamespace    = "ustc:courses"
           xmlns              = "ustc:courses"
           elementFormDefault = "qualified">

<xs:element name="course">
  <xs:complexType>
    <xs:sequence>

      <xs:element name="units" type="xs:int"/>

      <xs:element name="teachers">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="teacher" minOccurs="1"
                maxOccurs="unbounded">
              <xs:complexType>
                <xs:attribute name="familyName"   type="xs:string"/>
                <xs:attribute name="personalName" type="xs:string"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <xs:element name="students">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="student" minOccurs="1"
                maxOccurs="unbounded">
              <xs:complexType>
                <xs:attribute name="studentid" type="xs:string"/>
                <xs:attribute name="score"     type="xs:float"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>

    </xs:sequence>

    <xs:attribute name="courseName" type="xs:string" />
  </xs:complexType>
</xs:element>

</xs:schema>
```

Example: Course XSD / Usage

- When using the new schema, we need to reference it from within the XML document

Listing: The corresponding part of the courseWithNamespace.xml file

```
<course courseName="Distributed␣Computing"
        xmlns="ustc:courses">
```

Listing: The full courseWithNamespace.xml file

```xml
<?xml version="1.0" encoding="UTF-8"?>

<course courseName="Distributed␣Computing"
        xmlns="ustc:courses">

  <units>60</units>

  <teachers>
    <teacher familyName="Weise" personalName="Thomas" />
    <teacher familyName="Chen"  personalName="Xianglan" />
  </teachers>

  <students>
    <student studentid="SA11111111" score="85.5" />
    <student studentid="SA22222222" score="73.0" />
    <student studentid="SA33333333" score="90.0" />
  </students>

</course>
```

Example: Course XSD / Usage

- We also need to tell an XML parser where to find the schema file if we want to enable the parser to actually check the XML file for validity

Listing: The corresponding part of the courseWithNamespaceAndSchemaLocation.xml file

```
<course courseName="Distributed␣Computing"
        xmlns="ustc:courses"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="ustc:courses␣
            https://raw.githubusercontent.com/thomasWeise/distributedComputingExamples/master/xml/xml/course
```

Example: Course XSD / Usage

- We also need to tell an XML parser where to find the schema file if we want to enable the parser to actually check the XML file for validity
- The `schameLocation` attribute assigns a URL for download to the schema URI.

Listing: The corresponding part of the courseWithNamespaceAndSchemaLocation.xml file

```
<course courseName="Distributed␣Computing"
        xmlns="ustc:courses"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="ustc:courses␣
           https://raw.githubusercontent.com/thomasWeise/distributedComputingExamples/master/xml/xml/course
```

Example: Course XSD / Usage

- We also need to tell an XML parser where to find the schema file if we want to enable the parser to actually check the XML file for validity
- The `schameLocation` attribute assigns a URL for download to the schema URI.
- I just use the GitHub location of my XSD file...

Listing: The corresponding part of the courseWithNamespaceAndSchemaLocation.xml file

```
<course  courseName="Distributed␣Computing"
         xmlns="ustc:courses"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="ustc:courses␣
             https://raw.githubusercontent.com/thomasWeise/distributedComputingExamples/master/xml/xml/course
```

Listing: The full courseWithNamespaceAndSchemaLocation.xml file

```xml
<?xml version="1.0" encoding="UTF-8"?>

<course courseName="Distributed␣Computing"
        xmlns="ustc:courses"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="ustc:courses␣
           https://raw.githubusercontent.com/thomasWeise/distributedCompu

  <units>60</units>

  <teachers>
    <teacher familyName="Weise" personalName="Thomas" />
    <teacher familyName="Chen"  personalName="Xianglan" />
  </teachers>

  <students>
    <student studentid="SA11111111" score="85.5" />
    <student studentid="SA22222222" score="73.0" />
    <student studentid="SA33333333" score="90.0" />
  </students>

</course>
```

XSD Features

- We can now *validate* an XML document against our schema

XSD Features

- We can now *validate* an XML document against our schema
- We can do this locally, or by using online validators, e.g., `http://xsdvalidation.utilities-online.info/`, `http://www.xmlforasp.net/schemavalidator.aspx`

XSD Features

- We can now *validate* an XML document against our schema
- We can do this locally, or by using online validators, e.g., `http://xsdvalidation.utilities-online.info/`, `http://www.xmlforasp.net/schemavalidator.aspx`
- For every XML application, there should be a schema

XSD Features

- We can now *validate* an XML document against our schema
- We can do this locally, or by using online validators, e.g., `http://xsdvalidation.utilities-online.info/`, `http://www.xmlforasp.net/schemavalidator.aspx`
- For every XML application, there should be a schema
- The schema tell us whether an XML document has the correct structure

XSD Features

- Schema definitions have lots of more features that are our of the scope of this lesson

XSD Features

- Schema definitions have lots of more features that are our of the scope of this lesson, such as
  - re-usable data type definitions (`complexType` , `simpleType`)
  - hierarchical type definitions with inheritance and extensions or restrictions

XSD Features

- Schema definitions have lots of more features that are our of the scope of this lesson, such as
  - re-usable data type definitions (`complexType`, `simpleType`)
  - hierarchical type definitions with inheritance and extensions or restrictions
  - optional elements (`choice`)

XSD Features

- Schema definitions have lots of more features that are our of the scope of this lesson, such as
  - re-usable data type definitions (`complexType`, `simpleType`)
  - hierarchical type definitions with inheritance and extensions or restrictions
  - optional elements (`choice`)
  - `group`s of elements or attributes

XSD Features

- Schema definitions have lots of more features that are our of the scope of this lesson, such as
  - re-usable data type definitions (`complexType`, `simpleType`)
  - hierarchical type definitions with inheritance and extensions or restrictions
  - optional elements (`choice`)
  - `group`s of elements or attributes
  - documentation
  - ...

Summary

- XML dialects can be specified with XML Schemas [1–3, 7]

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://www.it-weise.de

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, "Der Wanderer über dem Nebelmeer", 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn, editors. *XML Schema Part 1: Structures Second Edition*. W3C Recommendation. MIT/CSAIL (USA), ERCIM (France), Keio University (Japan): World Wide Web Consortium (W3C), October 28, 2004. URL `http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/`.
2. Paul V. Biron and Ashok Malhotra, editors. *XML Schema Part 2: Datatypes Second Edition*. W3C Recommendation. MIT/CSAIL (USA), ERCIM (France), Keio University (Japan): World Wide Web Consortium (W3C), October 28, 2004.
3. Cliff Binstock. *The XML Schema – Complete Reference*. Reading, MA, USA: Addison-Wesley Professional, 2003. ISBN 0672323745 and 9780672323744. URL `http://books.google.de/books?id=pCLJly0eZwEC`.
4. Marco Skulschus and Marcus Wiederstein. *XML Schema*. Berlin, Germany: Comelio GmbH, 2008. ISBN 3898424723, 393970122X, 9783898424721, and 9783939701224. URL `http://books.google.de/books?id=Vi6dTuL6g7kC`.
5. Eric Van der Vlist. *XML Schema – The W3C's Object-Oriented Descriptions for XML*. O'Reilly Series. Sebastopol, CA, USA: O'Reilly Media, Inc., 2002. ISBN 0596002521 and 9780596002527. URL `http://books.google.de/books?id=2h8T6xqKeVUC`.
6. Jack Lindsey. Subtyping in w3c xml schema, part 1 – three degrees of inheritance. *The Data Administration Newsletter – TDAN.com*, 12(4), April 3, 2008. URL `http://www.tdan.com/view-articles/7185`.
7. David C. Fallside and Priscilla Walmsley, editors. *XML Schema Part 0: Primer Second Edition*. W3C Recommendation. MIT/CSAIL (USA), ERCIM (France), Keio University (Japan): World Wide Web Consortium (W3C), second edition, October 28, 2004. URL `http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/`.