XML DOCUMENTS, DTD, AND XML SCHEMA

Hasan Faik ALAN

CMPE 422

18.11.2011

Content

- XML Documents
  - Well-form and Validity of XML Documents
  - Schema Languages
  - Document Type Definition (DTD)
- XML Schema
  - What is XML Schema?
  - Syntax
  - Expressive Power, Database Constraints

Well-formed XML

A Well-formed XML Document must have:

- Version of XML being used

  <?xml version="1.0" encoding="UTF-8" **?>**
- Single root element

  <project>

  <….>

  ….

  </..>

  </project>
- Matching pair of start and end tags, correctly nested

  <Worker>

  </Worker>

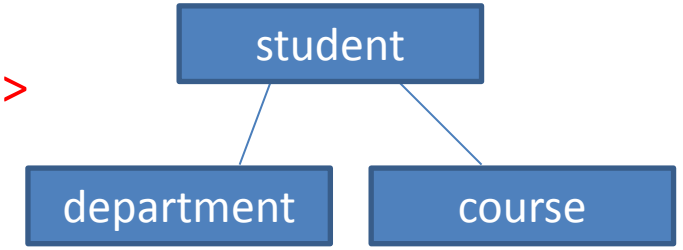
Purpose of Well-Formed XML

A well formed XML document is syntactically correct.

Well-form Allows,

- Traversing a document
- Creating an internal tree representation

```
<student>
        <department>CMPE</department>
        <course>422</course>
</student>
```



Validity of an XML Document

- Requirements
  - XML Document must contain a reference to a DTD or XML schema file
  - All the elements and attributes used in XML are declared in the referenced file
  - XML document must follow the rules declared in the referenced file (e.g. nested structure)
- XML document is valid if
  - The document is well formed
  - Follows from the DTD or XML Schema file

Validity of an XML Document

- XML Processors

  Use the referenced schema language to validate XML document

  - Validating
  - Non-validating

- A validating processor must be able to report discovered validity errors, but may continue processing.
- Must read every piece of a document and report all well-formedness and validity violations.

Well-from & Validity

```
<?xml version="1.0"?>
<greeting>Hello, world!</greeting>
```

- Above XML Document is Well-formed but not Valid.
- No document type declaration, no compliance to validate

Definition: An XML document is **valid** if it has an associated document type declaration and if the document complies with the constraints expressed in it. (w3c)

Schema Languages

- Document Type Definition(DTD)
  - The oldest schema language for xml
  - Still used in many applications because of its ubiquity
- XML Schema
  - The successor of DTDs
  - More powerful than DTDs in describing XML languages
  - Rich Datatyping system, detailed constraints on logical structure
- Other
  - Schematron: uses XPATH expressions, checks presence of XML patterns
  - DSDL (Document Schema Description Languages): many small schema languages

Document Type Definition(DTD)

Definition: The XML **document type declaration** contains or points to markup declarations that provide a grammar for a class of documents. (W3C)

- The oldest schema language for xml
- Supported widely because of inclusion in XML 1.0 standard
- Still used in many applications because of its ubiquity
- Can be embedded into XML Documents
- Has its own syntax other than XML, requires specialized processor

Document Type Definition(DTD)

- A valid XML Document with DTD
  - Has doctype declaration
  - Compliance with schema specified

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
<!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world!</greeting>
```

Document Type Definition(DTD)

example.dtd:

<! ELEMENT people_list (person)*>

<! ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>

<! ELEMENT name (#PCDATA)>

<! ELEMENT birthdate (#PCDATA)>

<! ELEMENT gender (#PCDATA)>

<! ELEMENT socialsecuritynumber (#PCDATA)>

Document Type Definition(DTD)

A valid XML file using example.dtd file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
        <person>
                <name> H.Faik ALAN </name>
                <birthdate> 12-07-1989 </birthdate>
                <gender>Male</gender>
        </person>
</people_list>
```

Document Type Definition(DTD)

DTD Syntax

- Symbols following an element means repeat that element:
  - ***** zero or more times
  - **+** one or more times
  - **?** zero or one times

  Without any of the above characters element appears exactly once

- **#PCDATA** (similar to a string data type) ,used in leaf nodes

- **If DTD is embedded into XML document**

  <?xml version="1.0" **standalone="yes"** ?>

  otherwise

  <?xml version="1.0" **standalone="no"** ?>
  <!DOCTYPE example SYSTEM "example.dtd">

Document Type Definition(DTD)

Syntax Overview

DTD embedded directly into XML file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
 <!DOCTYPE people_list [
<!ELEMENT people_list (person)*>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT socialsecuritynumber (#PCDATA)> ]>
<people_list>
        <person>
                <name>H.Faik ALAN</name>
                <birthdate>12-07-1989</birthdate>
                <gender>Male</gender>
                 <!– socialsecuritynumber omitted from person which is valid-->
        </person>
</people_list>
```

Document Type Definition(DTD)

Limitations of DTD

- No explicit support for namespaces
- DTD has a syntax other than XML thus requires specialized processors
- DTD elements are forced to follow specified ordering, unordered elements are not permitted.
- Lacks expressiveness

These drawbacks led to the development of **XML schema**

XML Schema

- A more general XML based alternative to DTD
- Also referred to as XML Schema Definition(XSD)
- XML Schema first became a W3C Recommendation in 02. May 2001 then in 28. Oct 2004
- As in XML DTD, XML Schema is based on the tree data model with elements and attributes
- Better support for document structure, attributes, and data-typing.
- Native namespace support

XML Schema

- **Sample XML Schema, “book”**

```
<?xml version="1.0" encoding="utf-8"?>
        <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <xs:element name="book">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="title" type="xs:string"/>
                        <xs:element name="author" type="xs:string"/>
                    </xs:sequence>
                    <xs:attribute name="isbn" type="xs:string"/>
                </xs:complexType>
            </xs:element>
</xs:schema>
```

XML Schema

- **XML file using the “book” schema**

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://example.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://example.com book.xsd"
isbn="0123747503">
        <title>Computer Organization and Design</title>
        <author>David A. Patterson</author>
</book>
```

XML Schema

- Schema Element
  - Opens the schema, root element of every schema
  - Can hold target name space and default options

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

Purpose:

Declares that elements and data types comes from "http://www.w3.org/2001/XMLSchema" namespace should be prefixed with **xs:**

XML Schema

- Target Namespace

  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://oursite.com" ….

Purpose:

Declares the namespace of the elements that will use the specified schema

- Default Namespace

  xmlns="http://oursite.com"

Purpose:

Declares the default namespace, all used elements in the xml file are declared in this namespace

XML Schema

- Referencing a Schema in an XML Document

```
<?xml version="1.0"?>
<book
xmlns="http://oursite.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://oursite.com book.xsd">
…
</book>
```

- **xmlns="http://oursite.com"**
  All used elements in the xml file are declared in = " http://oursite.com" namespace

- **xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**
  To specify schema location XML Schema Instance namespace is included

- **xsi:schemaLocation="http://oursite.com book.xsd"**>
  Specify the location of schema file

XML Schema

- Simple Types
  - Element that contains only text, does not contain any other element or attributes
  - Text can be one of the types included in the XML Schema definition (boolean, string, date, etc.), or a custom type
- Usage:

Simple Type:

XML: <age>36</age>

XSD : <xs:element name="age" type="xs:integer"/>

Attribute Usage:

XML: <title lang="EN">Sweet November</lastname>

XSD : <xs:attribute name="lang" type="xs:string"/>

XML Schema

- Complex Types
  - Contains other elements and attributes
- Usage:

XML:

```
<person>
  <firstname>Hasan</firstname>
  <lastname>ALAN</lastname>
</person>
```

XSD:

```
<xs:element name=“person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML Schema

- Data Types
  - String Data Type

    <surname>ALAN</surname>

    <xs:element name="surname" type="xs:string"/>
  - Date Data Type

    <finish>18-11-2011</finish>

    <xs:element name="finish" type="xs:date"/>
  - Numeric Data Type

    <grade>99.9</grade>

    <xs:element name= "grade" type="xs:decimal"/>
  - Boolean, float, double ..

XML Schema

- More on Syntax

XML Schema Part 1: Structures

http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Datatypes

http://www.w3.org/TR/xmlschema-2/

XML Schema

- Expressive Power, Database Constraints

Constraints

- Unique
- Primary Key
- Foreign Key

XML Schema

- **Constraints**

xsd:unique

Specifies elements that corresponds to unique attributes in a relational database that are not primary keys.

Usage: field of employeeDependent element which is of Dependent type must be unique

Impose unique constraint on employeeDependent element, find the element using xpath *

```
<xsd:unique name="dependentNameUnique">
        <xsd:selector xpath="employeeDependent"/>
        <xsd:field xpath="dependentName"/>
</xsd:unique>
```

Define the an element of type Dependent, dependentName field of its type will be unique

```
<xsd:element name="employeeDependent" type="Dependent"/>
```

Define the Dependent type which contains the dependentName field

```
<xsd:complexType name="Dependent">
        <xsd:sequence>
                <xsd:element name="dependentName" type="xsd:string"/>
                <xsd:element name="birthDate" type="xsd:date"/>
        <xsd:sequence>
</xsd:complexType>
```

*XPath is a language for finding information in an XML document. e.g. XQuery is built on Xpath expressions

XML Schema

- **Constraints**

  xsd:key

  Specifies elements that corresponds to primary key in a relational database.
  Usage: field of department element which is of Department type must be primary key

  Impose primary key on department element, find the element using xpath *

```
<xsd:key name="departmentNumberKey">
        <xsd:selector xpath="department"/>
        <xsd:field xpath="deparmentNumber"/>
</xsd:key>
```

  Define the an element of type Department, departmentName field of its type will be unique

```
<xsd:element name="department" type="Department"/>
```

  Define the Dependent type which contains the dependentName field

```
<xsd:complexType name="Department">
        <xsd:sequence>
                <xsd:element name="departmentName" type="xsd:string"/>
                <xsd:element name="departmentNumber" type="xsd:string"/>
        <xsd:sequence>
</xsd:complexType>
```

  *XPath is a language for finding information in an XML document. e.g. XQuery is built on Xpath expressions

XML Schema

- **Constraints**

  xsd:keyref

  Specifies the foreign key relation in a relational database.

  Usage: employeeDepartmentNumber field of employee element will be a foreign key that references departmentNumber

  Foreign key constraint on employee element's department number field,find the element using xpath *
  Foreign key will be referenced using departmentNumberKey element.

```
<xsd:key name="employeeDepartmentNumberKeyRef" refer="departmentNumberKey">
        <xsd:selector xpath="employee"/>
        <xsd:field xpath="employeeDepartmentNumber"/>
</xsd:key>
```

  Define the an element of type Employee, employeeDepartmentNumber field of its type will be primary key

```
<xsd:element name="employee" type="Employee"/>
```

  Define the depertmentNumberKey type which provides access to department number

```
<xsd:key name="departmentNumberKey">
        <xsd:selector xpath="department"/>
        <xsd:field xpath="deparmentNumber"/>
</xsd:key>
```

  *XPath is a language for finding information in an XML document. e.g. XQuery is built on Xpath expressions

Additional Examples

- Employee Info

Element	Constraints
Employee Info	
Employee	Zero or more times, Text
Name	Once, Text
Department	Once, Text
Employee Number	Once , Integer, Required Data

- Source:http://www.xmlmaster.org/en/article/d01/c04/

Additional Examples

- Employee Info
  - DTD File

employee.dtd:

<!ELEMENT Employee_Info (Employee)*>

<!ELEMENT Employee (Name, Department, Telephone, Email)> <!ELEMENT Name (#PCDATA)>

<!ELEMENT Department (#PCDATA)>

<!ELEMENT Telephone (#PCDATA)>

<!ELEMENT Email (#PCDATA)>

<!ATTLIST Employee Employee_Number CDATA #REQUIRED>

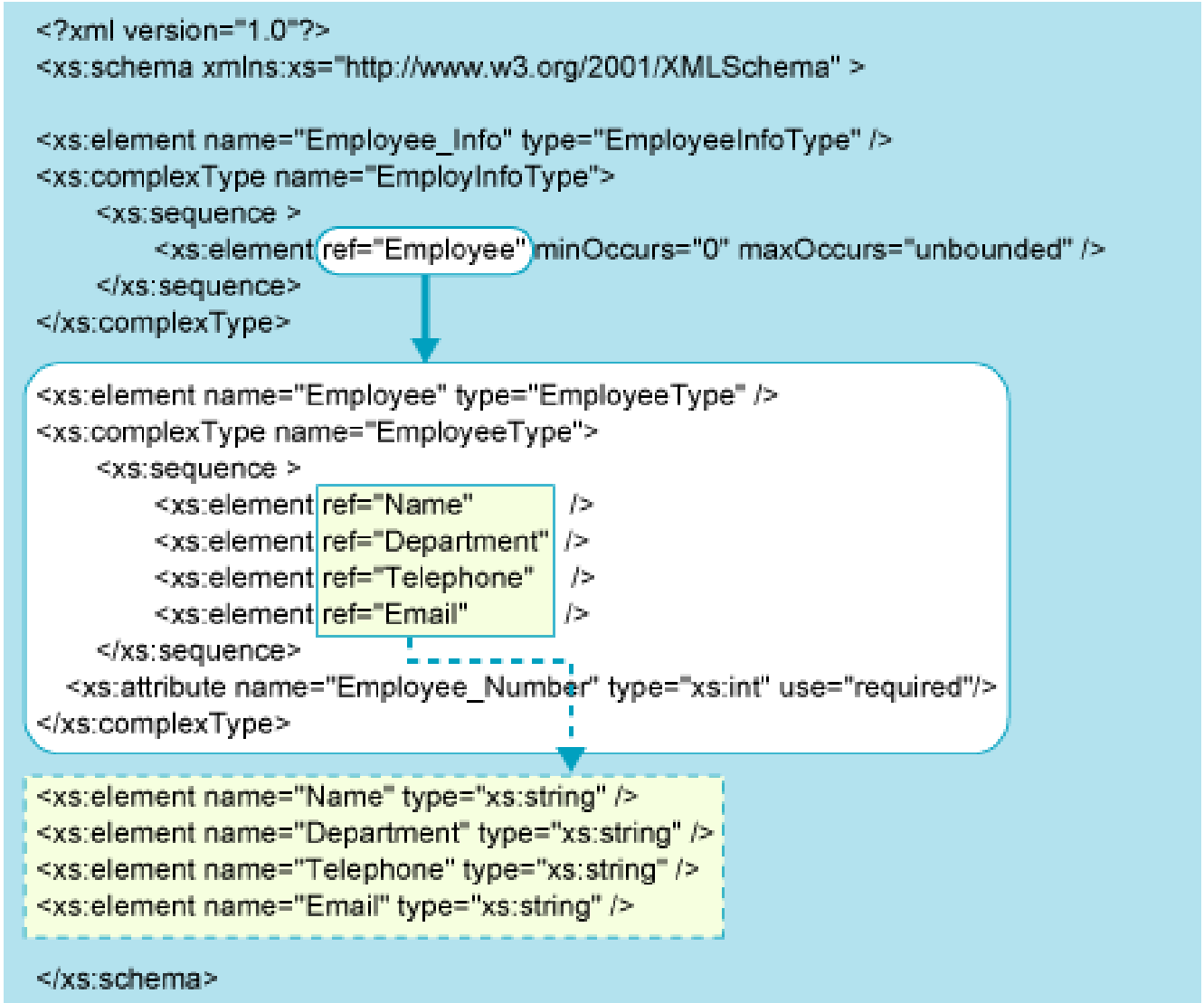
Additional Examples

- Employee Info
  - Schema File, employee.xs

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
                <xs:element name="Employee_Info" type="EmployeeInfoType" />
                        <xs:complexType name="EmployeeInfoType">
                                <xs:sequence>
                                    <xs:element ref="Employee" minOccurs="0" maxOccurs="unbounded" />
                                </xs:sequence>
                        </xs:complexType>
                <xs:element name="Employee" type="EmployeeType" />
                      <xs:complexType name="EmployeeType">
                            <xs:sequence >
                                <xs:element ref="Name" />
                                <xs:element ref="Department" />
                                <xs:element ref="Telephone" />
                                <xs:element ref="Email" />
                            </xs:sequence>
                                 <xs:attribute name="Employee_Number" type="xs:int" use="required"/>
                        </xs:complexType>
                 <xs:element name="Name" type="xs:string" />
                <xs:element name="Department" type="xs:string" />
                 <xs:element name="Telephone" type="xs:string" />
                 <xs:element name="Email" type="xs:string" />
</xs:schema>
```

Additional Examples

- Employee Info



Additional Examples

- Employee Info

```
<?xml version="1.0"?>
 <Employee_Info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://oursite.com employee.xs">
        <Employee Employee_Number="1">
                <Name>Hasan Faik ALAN</Name>
                <Department>Design Department</Department>
                <Telephone>0123456</Telephone>
                 <Email>asdf@ourcompany.com</Email>
        </Employee>
        <Employee
                Employee_Number=“2">
                <Name>Aziz Kerem</Name>
                <Department>Sales Department</Department>
                <Telephone></Telephone> <
                Email>azizol@ourcompany.com</Email>
        </Employee>
</Employee_Info>
```

References

- **Book:** Fundamentals of Database Systems, 5/E
  Ramez Elmasri
- http://www.w3.org/XML/Schema
- http://www.xmlmaster.org/en/article/d01/c04/
- http://www.w3schools.com

Next

- **XML representation of a relational database**