German Jordanian University الجامعة الألمانية الأردنية

CS355 Web Technologies

Dr. Ismail Hababeh

German-Jordanian University

Lecture 9

The XML Documents

- eXtensible Markup Language (XML) is a super set of HTML.
- XML Document is a universal format document type used for data exchange and data storage.
- XML documents must conform their metadata such as Document Type Definition (DTD) or Schema which specifies complex data types, elements (and sub-elements) and its attributes,.. etc.

The XML Documents

- XML allows user defined tags.
- XML Documents can be used to represent and transfer structured data in the hierarchy of element tags.
- XML is widely used for deployment descriptors and configuration specifications.

XML Document - Example

This simple XML code demonstrates *students* GPA record.

```
<?xml version = “1.0”?>
   <students>
        <student id=123>
              <name>John Smith</name>
              <gpa>3.5</gpa>
        </student>
        <student id=789>
              <name>Scott Tiger</name>
              <gpa>3.9</gpa>
        </student>
   </students>
```

Notes on the XML Document Example

- The tag "?" is a Processing Instruction (PI) to inform XML parser that this XML document must conform XML v. 1.0.
- This XML document shows two student records, each record consists of student name and his grade point average (gpa).
- The tag students is a top-level (root element) which has several sub-elements called student*t*.

Notes on the XML Document Example

- Each student element consists of two sub-elements called name and gpa.
- The identifier id in student tag is an attribute of element student.
- Every XML document must have its metadata:

  just like a data record in a database table, it must satisfy the definition of the table which is called *schema* in database.

XML Document Formats

- There exists different formats of XML document:
  - Internal Document Type Definition (DTD) within XML file.
  - External DTD (as an external reference)
  - Schema Document
- DTD itself does not use XML syntax and it is not very flexible.

Internal Document Type Definition DTD

- Elements are declared within the XML files.
- Standalone attribute in XML declaration must be set to yes.

Syntax:

<!DOCTYPE root-element [element-declarations]>

Internal DTD - Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE course [
  <!ELEMENT course (name,number,credits)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT number (#PCDATA)>
  <!ELEMENT credits (#PCDATA)>
] >
<course>
  <name>Web Technologies</name>
  <number>CS355</number>
  <credits>3</credits>
</course>
```

Internal DTD – Example Notes

- The first line is the XML document header.
- The second line is the document type declaration (referred to as the DOCTYPE: <!DOCTYPE course [
- The DOCTYPE declaration starts with an exclamation mark (!).
- The DOCTYPE informs the parser that a DTD is associated with this XML document.
- The DOCTYPE declaration is followed by DTD body, where elements, attributes, entities, and notations are declared.

Internal DTD – Example Notes

- <!ELEMENT name (#PCDATA)> defines the element name to be of type "#PCDATA" ; parse-able text data.
- The DTD declaration section is closed using]>.

Rules:

- The document type declaration must appear at the start of the document (preceded only by the XML header), it is not allowed anywhere else within the document.
- The element declarations must start with an exclamation mark.
- The name in the DTD must match the element type of the root element.

External DTD

- External DTD elements are declared outside the XML file.
- External DTD elements are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL.
- To refer the external DTD elements, standalone attribute in the XML declaration must be set to “no”. That means, elements declaration includes information from the external source.
- Syntax:

```
<!DOCTYPE root-element SYSTEM "file-name.dtd">
```

External DTD - Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!DOCTYPE course SYSTEM "course.dtd">

<course>

 <name>Web Technologies</name>

 <number>CS355</number>

 <credits>3</credits>

</course>
```

External DTD - Example

The content of the course.dtd file are:

```
<!ELEMENT course (name,number,credits)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT number (#PCDATA)>

<!ELEMENT credits (#PCDATA)>
```

XML Schemas

- XML Schema is known as XML Schema Definition (XSD).
- Used to describe and validate the structure and the content of XML data.
- Defines the elements, attributes and user data types.
- Defines sub-elements, sequence of child elements, fixed and default values of elements and attributes.
- Schema element supports Namespaces*.
- XSD is getting more popular and replacing DTD because schema itself is in XML format.
- Namespace: is a collection of XML elements and attributes identified by an International Resource Identifier (IRI)

XML Schema Document XSD

- XML Schema Document (XSD) is a metadata of an XML document.
- XSD specifies the syntax, structure, and constraints in a corresponding XML including data type or complex data type of elements, and attributes of elements.
- XSD Declaration Syntax:
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

XSD Document – Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">*
<xs:element name="contact">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="name" type="xs:string" />
                        <xs:element name="company" type="xs:string" />
                        <xs:element name="phone" type="xs:int" />
                </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

* indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace

XML Document – Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
<contact xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instancexsi:noNamespaceSchemaLocation="contact.xsd">*
    <name>Ahmad</name>
    <company>GJU</company>
    <phone>064294444</phone>
</contact>
```

* the XML document is an instance of XSD

XSD – Example 2

```
<?xml version=”1.0” encoding=”UTF-8”?>
    <xsd:schema xmlns:xsd=“http://www.w3.org/2001/XMLSchema”
            targetNamespace=http://www.myrecords.com
            xmlns=”http://www.myrecords.org”>
    <xsd:element name = “students”>
            <xsd:complexType>
                    <xsd:sequence>
                            <xsd:element ref=”student”
                               minOccurs=”1”
                               maxOccurs=”unbounded”/>
                    </xsd:sequence/>
            </xsd:complexType/>
    </xsd:element>
```

** UTF-8: Universal Coded Character Set + Transformation Format—**8**-bit*

```
<xsd:elememt name=”student”>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref=”name”/>
                <xsd:element ref=“gpa”/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name=”name” type=”xsd:string”/>
    <xsd:element name=”gpa” type=”xsd:float”/>
</xsd:schema>
```

Notes on the XSD Example

- The sequence tag in XSD Schema specifies the order and the occurrence of sub-elements in their parent element.
  - In the previous example; (name and gpa) are the sub-elements of the student object.
- The XMLNS plays the same role as package in Java and namespace in C++ to prevent naming collisions.
- An element name can be combined with a namespace prefix (xsd:). The idea of combining a namespace URL with a local name is to make any identifier name in XML universally unique.

XML DTD vs. XML Schema

XML DTD:

- There is no built-in data type.
- No new data type can be created
- The use of cardinality (# occurrences) is limited.
- Namespaces are not supported.
- Limited support for modularity and reuse.
- Can't put any restrictions on text content.
- Defaults for elements cannot be specified.
- Written in a non-XML format and are difficult to validate.

XML Schema:

- Provide much specification than DTDs.
- Support large number of built-in-datatypes.
- Namespace-aware.
- Extensible to future additions.
- Support the uniqueness.
- It is easier to define restrictions on data.

Comprehensive Examples
HTML – JAVA SCRIPT - XML

http://www.conta.uom.gr/conta/ekpaideysh/seminaria/xml/chap3.htm