Web Technologies Laboratory 02

Laboratory Continuous Assessment (LCA) [As per Rubrics]

Understanding of the Objective (5)	Performance (5)	Journal Submission and Ethics (Neatness, Handwriting, Timely submission) (5)	Orals (5)	Total (20)	Remarks	Instructor's Sign

Aim: Encode the given information using XML document. Construct an external DTD (Document Type Definition) for the XML document. Convert DTD to XML Schema and validate the document. Write a stylesheet to style the data using XSLT.

Objectives:

1. To understand XML
2. To learn validation of XML using DTD and XML Schema
3. To design a stylesheet to style the XML document

Theory:

1. XML
2. DTD and XML
3. XSLT

FAQ:

1. What is the difference between HTML and XML?
2. What are advantages of XML schema over the DTD?
3. What is meant by a well formed XML document?
4. How to validate a XML document using DTD or XML Schema?
5. What are XML namespaces and why are they used?
6. What is the use of eXtensible stylesheet and how to associate it with a XML document?
7. What do the minOccurs and maxOccurs attributes specify?

Output: Screenshots of the output to be attached.

eXtensible Markup Language (XML)

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine readable. It was was developed to overcome the limitations of html. It is used to represent and model data. XML tags are case-sensitive. Major components of XML: Elements, Attributes, Entities (Character, general, unparsed). Unlike html, Element tags in XML can be created by the user and they are not predefined by the language. Each document contains one root element. Attributes can provide parameters for an element. XML must be ***well-formed*** *(*correct syntax , tags match, tags nest, all characters legal)

Advantages of XML:

- Multiple delivery formats
- Information reuse
- Multiple sources of information
- Consistent presentation
- Reduce time spent fiddling with presentation

XML declaration at the start of XML file <?xml version="1.0" ?>

```
<?xml version=”1.0” ?>
<Customer Name=“Bill Smith” Age=“32” MaritalStatus=“Married”>
        <wife>
                <name>Angela</name>
                <age>25</age>
        </wife>
</customer>
```

Document Model:

A document model is used to enforce the structure within a document. Two types of document models can be used for XML validation. **Validation** is a process by which an XML document is validated.

- DTD – Document Type Definition
- XML Schema

DTD (Document Type Definition)

It is used to define the XML elements and attributes.It defines the relationships between different elements and attributes. Two types of DTD's exist:

Internal DTD: DTD exists as part of the XML document

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

<!DOCTYPE greeting [
                <!ELEMENT greeting (#PCDATA)>
]

>

<greeting>Hello, world!</greeting>
```

External DTD: DTD exist as an external file.External DTD is most common. You have to add a DOCTYPE declaration in the xml document before the root element as shown in the example below. Also the standalone attribute of the xml tag needs to be set to "no" to indicate it is depending on the external DTD for validation.

DOCTYPE Declaration contains following fields:

- !DOCTYPE
- Name of root element
- SYSTEM – tells parser that the DTD is external
- Location of DTD file

```
<!DOCTYPE books SYSTEM "book.dtd">
```

ELEMENT Declarations in DTD

- ANY – element can contain child elements or raw text data
- #PCDATA –Parsed Character data -Raw text
- Sequences (sequence of elements)
- Choices (a | b | c)
- Mixed Content (PC data and elements)
- EMPTY

The + suffix indicates that one or more of that element is required at that point

```
<!ELEMENT cataloging_info (abstract, keyword+)>
```

The * suffix indicates that zero, one, or more of that element is required at that point

```
<!ELEMENT catalog (category, cataloging_info, last_updated, copyright, maintainer, composer*, composition*)>
```

Suffixing an element name with a question mark (?) in the content model indicates that either 0 or 1 (but not more than one) of that element are expected at that position

<!ELEMENT composition (title, date, length?, instruments, description?, publisher?)>

A choice indicates one element or another but not both

- A choice is signified by a vertical bar |
- There can be two or more elements in a choice

<!ELEMENT date (year | ISODate)>

Mixed content is both #PCDATA and child elements in a choice, followed by an asterisk

<!ELEMENT description (#PCDATA | ul | cite)*>

Empty Elements – element does not need a closing tag

<!ELEMENT BR EMPTY>

ATTRIBUTE DECLARATION in DTD:

- <!ATTLIST
- element-name
- attr-name
- attr-type
- attr-default >

attr types:

- CDATA: any value is allowed
- (value | ….) enumeration of allowed values
- ID: must be a unique value within the current document
- IDREF, IDREFS: must match an already declared id (used to reference other elements in the document)

attr-default:

- **#REQUIRED:** The attribute must be explicitly provided
- **#IMPLIED:** Attribute is optional, no default provided
- **#FIXED "value" :** Attribute is set to this value always

XML Schema

XML Schema is commonly known as **XML Schema Definition (XSD)**. It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

- **XML Schema Features:**
    - Pattern matching
    - Rich set of data types
    - Attribute grouping
    - Supports XML namespaces
    - Follows XML syntax
- **The XML Schema specification consists of two parts:**
    - XML Schema: **Structures**. This specification consists of a definition language for describing and constraining the content of XML documents
    - XML Schema: **Datatypes**. This specification defines the datatypes to be used in XML schemas.

The XMLSchema language itself is a schema that is located at:

http://www.w3.org/2001/XMLSchema

Defining XMLSchema elements

- **Simple types**

  XML Schema has a lot of built-in data types. The most common types are:

  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

```
<xsd:element name=”fruit” type=”xsd:string”/>
```

  New simple types can be created by specifying values for one or more of the optional facets for the base type. Facets such as length, minLength , maxLength, minInclusive , maxInclusive, totalDigits , fractionDigits, pattern, enumeration

  Example

```
<xsd:simpleType name=“monthOfYear”>
        <xsd:restriction base=“xsd:integer”>
```

```
<xsd:minInclusive value="1" />
<xsd:maxInclusive value="12" />
</xsd:restriction>
</xsd:simpleType>
```

ComplexType

A complex type element may be considered to be one of four kinds according to the content that it contains:

- Element-only
- Text only
- Empty (with attributes)
- Mixed-content

  <xs:complexType mixed="true">

All elements defined in a complex type must be part of either: Sequence, Choice, Unordered group

Sequence

```
<xs:element name="note">
 <xs:complexType>
  <xs:sequence>
 <xs:element name="to" type="xs:string"/>
 <xs:element name="from" type="xs:string"/>
 <xs:element name="heading" type="xs:string"/>
 <xs:element name="body" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Choice: Only one element that is part of choice can appear once in a XML document

Example

```
<!-- create a complex type element called 'lang' -->
<xsd:complexType name="lang">
 <xsd:choice>
  <xsd:element name="english" type=xsd:string/>
<xsd:element name="italian" type=xsd:string/>
<xsd:element name="german" type=xsd:string/>
```

```
</xsd:choice>
</xsd:complexType>
```

Unordered group: Allows a set of elements to appear once within the XML document in any order

Example

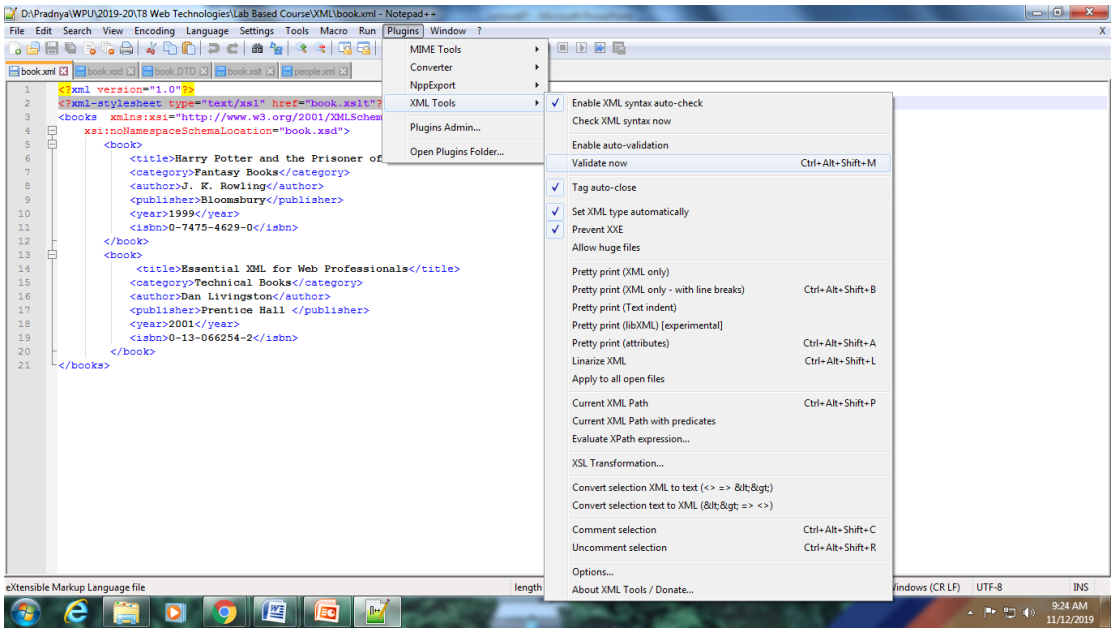
```
<!-- create a complexType element called 'activity' -->
<xsd:complexType name=”activity”>
  <xsd:any>
   <xsd:element name=”activity_type” type=xsd:string/>
 <xsd:element name=”category” type=xsd:string/>
 <xsd:element name=”length” type=xsd:duration/>
  </xsd:any>
</xsd:complexType>
```

Validating an XML document using Notepad++

Open the XML file which you want to validate in Notepad++

Either a DTD or XSD file needs to connected to the XML file using proper declaration.

Then go to Plugins->XMLTools->Validate now



XML Namespaces

As XML tags are created by the users and not defined by the language, there is a possibility of more than one user creating the tag with same name but meaning different depending on their requirement.

e.g. xyz company can create a tag <cost> where the cost of the product is including the GST

abc company might create a tag <cost> but here the cost data may be excluding the GST

Therefore we need some way of differentiating between the tags in the XML document created by different users.

This is achieved via namespaces. These are nothing but unique identifiers

Every XML document has a Namespace declaration at the top which looks like this:

xmlns:prefix=”sample”

xmlns:mark = “http://www.joe.com/ns/cost”

- URL (Uniform Resource Locator)
- URI (Uniform Resource Identifier)
- URN (Universal Resource Name)

The prefix attribute tells that all the tags in this namespaces will have the prefix sample. It is usually a practice to have the unique identifier which is URL, URI or URN as the mark attribute. It is not necessary that the URL or URI or URN even exist in the world.

Default Namespace: In many cases you will find that one namespace is being used most of the time. In such cases you can make that namespace default as follows

```
<myelement
        xmlns=“http://www.wpu.edu.in/pradnya/page1”
        xmlns:n1=“http://www.wpu.edu.in/pradnya/page2"
    <child1>
        I am from default namespace some data </child1>
    <n1:child1>
        I am from the other namespace
     </n1:child1>
</myelement>
```

XSL (eXtensible Stylesheet Language)

It is a styling language for XML. XSLT stands for XSL Transformations. It is used to transform the XML document into a HTML document which can then be rendered. XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically (html, pdf or another XML document). It is far more powerful than CSS. **XSLT style sheet is an XML document and as such must have a root element and an XML declaration**

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

<xsl:stylesheet version="1.0"

xmlns:xsl=http://www.w3c.org/1999/XSL/Transform >

<!-- style sheet rules here -->

</xsl:stylesheet>

Template is formed and the source xml tree is traversed starting from the root node

<xsl:value-of select="."/> Returns value of current node

<xsl:value-of select=".."/> Returns parent

<xsl:value-of select="blah"/> Returns value of "blah" (assuming "blah" is a child of the current node)

<xsl:value-of select ="../sibling"/> Returns value of "sibling" (assuming "sibling" is a sibling of the current node).

EXAMPLE of XML, DTD, XSD and XSLT

Book.xml with external DTD for validation

```
<?xml version="1.0" standalone="no"?>
<books>
    <book>
                <title>Harry Potter and the Prisoner of Azkaban</title>
                <category>Fantasy Books</category>
                <author>J. K. Rowling</author>
                <publisher>Bloomsbury</publisher>
                <year>1999</year>
                <isbn>0-7475-4629-0</isbn>
        </book>
        <book>
                <title>Essential XML for Web professionals</title>
                <category>Technical Books</category>
                <author>Dan Livingston</author>
                <publisher>Prentice Hall </publisher>
                <year>2001</year>
                <isbn>0-13-066254-2</isbn>
              </book>
</books>
```

Corresponding Document Type Definition (book.dtd)

```
<!ELEMENT books (book*)>
<!ELEMENT book (title,category,author,publisher,year,isbn)>
```

```
<!ELEMENT title (#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
```

XML Schema document (book.xsd) corresponding to book.xml

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="books">
   <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" ref="book"/>
    </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

<xsd:element name="book">
  <xsd:complexType>
            <xsd:sequence>
                    <xsd:element name="title" type="xsd:string" />
                    <xsd:element name="category" type="xsd:string" />
                    <xsd:element name="author" type="xsd:string" />
                    <xsd:element name="publisher" type="xsd:string" />
                    <xsd:element name="year" type="xsd:positiveInteger" />
                    <xsd:element name="isbn" type="xsd:string" />
            </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

eXtensible Style sheet (book.xslt) associated with the book.xml

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
 <html>
 <body>
  <h2>My Book Collection</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Category</th>
            <th>Author</th>
```

```
                <th>Publisher</th>
                <th>Year</th>
                <th>Isbn</th>
            </tr>
            <xsl:for-each select="books/book">
              <tr>
                <td><xsl:value-of select="title"/></td>
                <td><xsl:value-of select="category"/></td>
                        <td><xsl:value-of select="author"/></td>
                        <td><xsl:value-of select="publisher"/></td>
                <td><xsl:value-of select="year"/></td>
                        <td><xsl:value-of select="isbn"/></td>
              </tr>
            </xsl:for-each>
          </table>
        </body>
        </html>
      /xsl:template>

</xsl:stylesheet>
```

In order to attach a particular DTD to a XML document add the following line before the root element in xml file

```
<!DOCTYPE books SYSTEM "book.dtd">
```

In order to associate a xml schema file to the XML document add the xsd file as attribute value of xsi:noNameSpaceSchemaLocation of the root element in xml file as follows

```
<books xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="book.xsd">
```

In order to associate a xslt stylesheet to the xml document add following line before the root element in xml file

```
<?xml-stylesheet type="text/xsl" href="book.xslt"?>
```