ASSIGNMENT NO - 03

AIM: Design the XML document to store the information of the employees of any business organization and demonstrate the use of:

a) DTD
b) XML Schema

And display the content in (e.g., tabular format) by using CSS/XSL.

LEARNING OBJECTIVES:

To understand implementation of XML and DTD

THEORY:

INTRODUCTION TO XML:

XML is a software- and hardware-independent tool for storing and transporting data.

WHAT IS XML?

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

XML DOES NOT DO ANYTHING

Maybe it is a little hard to understand, but XML does not DO anything. This note is a note to John from Jani, stored as XML:

```
<note>
<to>John</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The XML above is quite self-descriptive:

- It has sender information.
- It has receiver information
- It has a heading
- It has a message body.

But still, the XML above does not DO anything. XML is just information wrapped in tags. Someone must write a piece of software to send, receive, store, or display it:

NOTE

To: John From: Jani *REMINDER*

Don't forget me this weekend!

THE DIFFERENCE BETWEEN XML AND HTML

XML and HTML were designed with different goals:

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags

XML DOES NOT USE PREDEFINED TAGS

The XML language has no predefined tags.

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

HTML works with predefined tags like <p>, <h1>, <table>, etc.

With XML, the author must define both the tags and the document structure.

XML IS EXTENSIBLE

Most XML applications will work as expected even if new data is added (or removed). Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <body>). Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.

The way XML is constructed, older version of the application can still work:

<note>
<date>2015-09-01</date>
<hour>08:30</hour>
<to>John</to>
<from>Jani</from>
<body>Don't forget me this weekend!</body>
</note>

OLD VERSION

NOTE

To: John From: Jani *HEAD: (NONE)*

Don't forget me this weekend!

NEW VERSION

NOTE

To: John

From: Jani

Date: 2015-09-01 08:30

Don't forget me this weekend!

XML SIMPLIFIES THINGS

- It simplifies data sharing
- It simplifies data transport
- It simplifies platform changes
- It simplifies data availability

Many computer systems contain data in incompatible formats. Exchanging data between incompatible systems (or upgraded systems) is a time-consuming task for web developers. Large amounts of data must be converted, and incompatible data is often lost.

XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data.

XML also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

With XML, data can be available to all kinds of "reading machines" like people, computers, voice machines, news feeds, etc.

XML Schema:

What is XML schema?

XML schema is a language which is used for expressing constraint about XML documents. There are so many schema languages which are used now days for example Relax- NG and XSD (XML schema definition).

An XML schema is used to define the structure of an XML document. It is like DTD but provides more control on XML structure.

Checking Validation

An XML document is called "well-formed" if it contains the correct syntax. A well-formed and valid XML document is one which has been validated against Schema.

XML Schema Example:

employee.xsd

```
1. <?xml version="1.0"?>
2. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3. targetNamespace="http://www.javatpoint.com"
4. xmlns="http://www.javatpoint.com"
5. elementFormDefault="qualified">
6. <xs:element name="employee">
7.   <xs:complexType>
8.     <xs:sequence>
9.       <xs:element name="firstname" type="xs:string"/>
10.      <xs:element name="lastname" type="xs:string"/>
11.      <xs:element name="email" type="xs:string"/>
12.    </xs:sequence>
13.  </xs:complexType>
14. </xs:element>
15. </xs:schema>
```

employee.xml

```
1. <?xml version="1.0"?>
2. <employee
3. xmlns="http://www.javatpoint.com"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://www.javatpoint.com employee.xsd">
6.   <firstname>vimal</firstname>
7.   <lastname>jaiswal</lastname>
8.   <email>vimal@javatpoint.com</email>
9. </employee>
```

Description of XML Schema:

<xs:element name="employee"> : It defines the element name employee.

<xs:complexType> : It defines that the element 'employee' is complex type.

<xs:sequence> : It defines that the complex type is a sequence of elements.

<xs:element name="firstname" type="xs:string"/> : It defines that the element 'firstname' is of string/text type.

<xs:element name="lastname" type="xs:string"/> : It defines that the element 'lastname' is of string/text type.

<xs:element name="email" type="xs:string"/> : It defines that the element 'email' is of string/text type.

XML Schema Data types:

There are two types of data types in XML schema.

1. simpleType
2. complexType

simpleType:

The simpleType allows you to have text-based elements. It contains less attributes, child elements, and cannot be left empty.

complexType:

The complexType allows you to hold multiple attributes and elements. It can contain additional sub elements and can be left empty.

XML – DTD:

The XML Document Type Declaration, commonly known as DTD, is a way to describe XML language precisely. DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language.

Syntax:

```
<!DOCTYPE element DTD identifier
[
  declaration1
  declaration2
  .....
]>
```

In the above syntax:

- The **DTD** starts with <!DOCTYPE delimiter.
- An **element** tells the parser to parse the document from the specified root element.
- **DTD identifier** is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called **External Subset.**
- **The square brackets [ ]** enclose an optional list of entity declarations called *Internal Subset*.

Internal DTD:

A DTD is referred to as an internal DTD if elements are declared within the XML files. To refer it as internal DTD, *standalone* attribute in XML declaration must be set to **yes**. This means, the declaration works independent of an external source.

Syntax:

Following is the syntax of internal DTD −

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

where *root-element* is the name of root element and *element-declarations* is where you declare the elements.

Example:

Following is a simple example of internal DTD −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address [
  <!ELEMENT address (name,company,phone)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT company (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
]>
```

```
<address>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</address>
```

External DTD:

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal *.dtd* file or a valid URL. To refer it as external DTD, *standalone* attribute in the XML declaration must be set as no. This means, declaration includes information from the external source.

Syntax:

Following is the syntax for external DTD −

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

where *file-name* is the file with *.dtd* extension.

Example:

The following example shows external DTD usage −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</address>
```

The content of the DTD file **address.dtd** is as shown −

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

CONCLUSION: Hence we design the XML document which stores the employee information and displaying the information in tabular form.