Experiment No. 14

Title: Creating valid XML document with Internal DTD and External DTD.

Objective:

At the end of practical session student will be able to create valid XML document with internal DTD and External DTD for given requirement.

Theory:

Markup: -A system for marking or tagging content that indicates its logical structure and provides for special computer processing of that content.

XML stands for extensible markup language. It is a meta- language that describes other markup language. It allows creating new tags that reflects the meaning of the contents it encloses. XML is a method for putting structured data in a text file. Thus Xml gives meaning to data or simple text. HTML is also a markup language but it is designed only to format the text and doesn't impart any meaning to the text it markups.

XML gives its adopters:

- Interoperability
- vendor-neutrality
- platform-independence
- improved data search

Structure of XML document:

Every xml document is divided into two parts: 1. prolog , 2.document instance.

1. **Prolog:**

The prolog refers to the information that appears before the start tag of the document or root element. It includes information that applies to the document as a whole, such as character encoding, document structure, and style sheets.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="show_book.xsl"?>
<!DOCTYPE catalog SYSTEM "catalog.dtd">
<!--catalog last updated 2000-11-01-->
```

Processing instructions can also appear in the prolog, for example, the xml-stylesheet processing instruction,

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

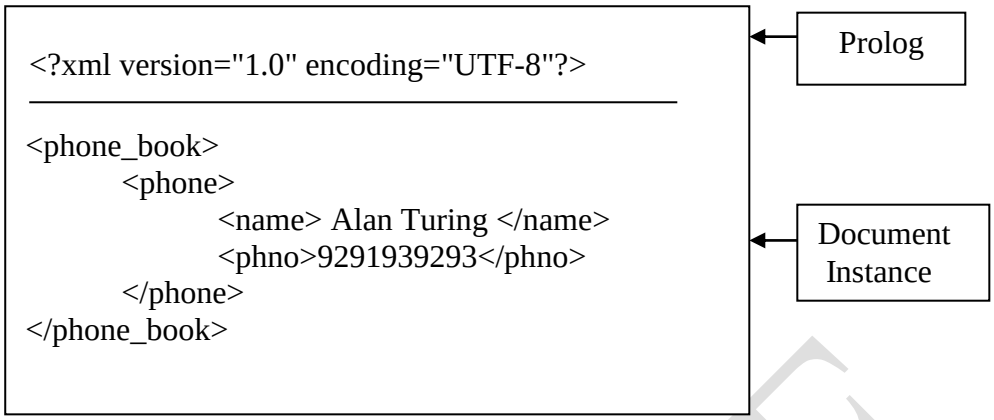
Comments can appear in the prolog, for example,

```
<! — catalog last updated 2000-11-01--!>.
```

2. **Document Instance:**

Document instance follows the prolog. It contains actual document data organized as a hierarchy of elements. The document is enclosed within root element or document element. The data in the document is markup with the elements, attributes and other components of XML.

Example:



Components of XML:

There are six type of markup defined in XML. They are:

1.Elements
2.Attributes
3.Comments
4.Processing Instructions
5.Entity reference
6.CDATA section

1. Element : A logical structure in an XML document that is delimited by a start and an end tag. An element encloses either text data or other sub-elements or both.

e.g. <book> The XML Handbook </book>

Properties of elements:

- Elements must have a closing tag.
- Elements must be properly nested
- Elements can have different content types.

An element can have ***element*** content, ***mixed*** content, ***simple*** content, or ***empty*** content. An element can also have ***attributes***.

XML elements must follow these naming rules:

- Names can contain letters, numbers, and other characters.
- Names must not start with a number or punctuation character.
- Names must not start with the letters xml (or XML, or Xml, etc).
- Names cannot contain spaces

2.Attributes:

Attributes are used to provide additional information about elements. Attributes are specified in the start tag of xml element, just like HTML. they often provide information that is not a part of the data. Attributes are specified in name value pair and the values are specified either in single quotes or double quotes.

Example:

<book media = ”CD” > The XML Handbook </book>
<book media = ‘CD’ > The XML Handbook </book>

If the attribute value itself contains double quotes it is necessary to use single quotes and if the attribute value itself contains single quotes it is necessary to use double quotes.

Some of the problems with using attributes are:

- Attributes cannot contain multiple values (child elements can).
- Attributes are not easily expandable (for future changes).
- Attributes cannot describe structures (child elements can).
- Attributes are more difficult to manipulate by program code.
- Attribute values are not easy to test against a Document Type Definition (DTD) - which is used to define the legal elements of an XML document

If you use attributes as containers for data, you end up with documents that are difficult to read and maintain. Try to use elements to describe data. Use attributes only to provide information that is not relevant to the data.

3. Comment:

XML comments are used to include explanatory and descriptive notes in a document. Comments are ignored by *XML* parsers and may appear anywhere in a document outside other *XML* markup. *XML* comments look very much like *HTML* comments.

As in *HTML*, you start a comment with <!-- and end it with -->. Here's an example:

<!--The next element contains a heading.-->

4. Processing Instructions:

Processing instructions (PIs) allow documents to contain instructions for applications.

A processing instruction is a bit of information meant for the application using the XML document. That is, they are not really of interest to the XML parser. Instead, the instructions are passed intact straight to the application using the parser. The application can then pass this on to another application or interpret it itself.

All processing instructions follow the generic format of

<?NAME_OF_APPLICATION_INSTRUCTION_IS_FOR INSTRUCTIONS?>

The only restriction here is that you can't use <?xml?> (or <?XML?>, which is also reserved). Examples:

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

<?xml-stylesheet type="text/css" href="ch01_04.css"?>

Note: processing instructions like these one are not built into *XML*, but have been agreed upon by various browser manufacturers.

You might have PIs something like:

<?JAVA_OBJECT JAR_FILE = "/java/myjar.jar"?>

5. Entity reference:

An XML document may consist of one or many storage units. These are called entities. Entities are essentially aliases that allow you to refer to large sections of text without having to type them out every time you want to use them. Entities may be either parsed or unparsed. The contents of a parsed entity are referred to as its replacement text; this text is considered an integral part of the document.

An unparsed entity is a resource whose contents may or may not be text, and if text, may be other than XML. Each unparsed entity has an associated notation, identified by name.

Beyond a requirement that an XML processor make the identifiers for the entity and notation available to the application, XML places no constraints on the contents of unparsed entities.

Parsed entities are invoked by name using entity references; unparsed entities by name, given in the value of ENTITY or ENTITIES attributes.

An entity reference refers to the content of a named entity. References to parsed general entities use ampersand (&) and semicolon (;) as delimiters.

Internal Entities :

If the entity definition is an Entity Value, the defined entity is called an internal entity. There is no separate physical storage object, and the content of the entity is given in the declaration. An internal entity is a parsed entity.

Example of an internal entity declaration:

```
<!ENTITY Pub-Status "This is a pre-release of the specification.">
<!ENTITY copyright "(c)2005 Don't copy without permission.">
```

here the general entity reference **copyright** is declared and when the entity reference **&copyright**; is used in *XML* document the, *XML* processor will replace it with "(c)2005 Don't copy without permission".

6.CDATA section:

CDATA sections may occur anywhere character data may occur; they are used to escape blocks of text containing characters, which would otherwise be recognized as markup. CDATA sections begin with the string "<![CDATA[" and end with the string "]]>"

Within a CDATA section, only the **CDEnd** string is recognized as markup, so that left angle brackets and ampersands may occur in their literal form; they need not (and cannot) be escaped using "<" and "&. CDATA sections cannot nest.

Example:

```
<![CDATA[
        var temperature
        temperature = 234.77
            if (temperature < 32) {
                    document.writeln("Below freezing!")
        }
]]>
```

Document Type Definition (DTD)

The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements. Thus DTDs written in a formal syntax that explains precisely which elements and entities may appear where in the document and what the elements' contents and attributes are.

A validating parser compares a document to its DTD and lists any places where the document differs from the constraints specified in the DTD. The program can then decide what it wants to do about any violations. Some programs may reject the document. Others may try to fix the document or reject just the invalid element. Validation is an optional step in processing XML. If XML document follows constraints specified in DTD completely then such XML document is called as valid XML document.

Internal DTD

If the DTD is declared inside the XML file, called as Internal DTD, it must be wrapped inside the <!DOCTYPE> as follows-

```
<!DOCTYPE personinfo [
<!ELEMENT person (name, profession)>
<!ELEMENT name (first_name, last_name)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT profession (#PCDATA)>]>
```

- !DOCTYPE personinfo defines that the root element of this document is personinfo
- !ELEMENT person (name) defines that the person (name) element must contain two elements: "name, profession" (“first_name, last_name”)
- !ELEMENT to defines the first_name (last_name, profession) element to be of type "#PCDATA" (#PCDATA, #PCDATA)

External DTD

If the DTD is declared outside the XML document in other file having .dtd extension, called as external DTD, the <!DOCTYPE> definition must contain a reference to the DTD file as shown below.

```
<?xml version="1.0"?>
<!DOCTYPE personinfo SYSTEM "personinfo.dtd">
<personinfo>
  <name>
        <first_name>Alan</first_name>
        <last_name>Turing</last_name>
  </name>
  <profession>Computer Scientist</profession>
</personinfo>
```

Contents of DTD files are as follows:

```
<!ELEMENT person (name, profession)>
<!ELEMENT name (first_name, last_name)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT profession (#PCDATA)>]>
```

If same DTD is used on multiple XML documents for validation external DTD is preferred while to increase the speed of validation many XML document uses internal DTD. Thus based on requirement internal DTD or external DTD is used.

Key concepts: Markup, XML, prolog, xml components, Document type definition.

Algorithm:

- Analyze the information inside the document.
- Define the structure, and the content.
- Define the markup to be used and declare it in DTD.
- Create the document with the defined markup.
- Validate XML document across DTD