Lecture 1

Contents: Introduction; Editors for writing XML document; Installing notepad++ editor; Writing the first XML document; Features of XML; Advantages of XML; XML versus HTML; Applications of XML; Exercises

1.1 Introduction

Extensible Markup Language (XML) is a markup language and special file format for storing, transmitting, and reconstructing data. It has a set of rules for encoding documents in a format that is readable by both human and machine. XML is used commonly for interchanging data over the Internet. In the context of markup languages, we shall also learn about two important languages viz., SGML and HTML.

Standard Generalized Markup Language (SGML) is a standard for defining generalized markup languages for documents. It uses markups to describe structure of the document. Markups rigorously define an object to perform processing. It was developed by *International Organization for Standardization* (ISO). SGML was reworked in 1998 into XML. Every SGML file has extension .sgml. An SGML document may have three parts:

1. SGML declaration
2. Prologue, containing a DOCTYPE declaration with the various markup declarations that together make a *Document Type Definition* (DTD)
3. Instance, containing one top-most element and its contents.

An SGML document may have many entities. SGML is mother of all markup languages.

HyperText Markup Language (HTML) is the standard markup language for documents designed to be displayed using a web browser. An HTML document has extension .html. It was developed by *Web Hypertext Application Technology Working Group* (WHATWG), a community of people interested in evolving HTML and related technologies, and released initially in 1993. It was extended from SGML. A web browser receives an HTML document from a web server or from local storage and renders the document into multimedia web pages. HTML describes the structure of a web page semantically.

XML was designed to be fully compatible with SGML. Any document that follows XML's syntax rules is, by definition, also following SGML's syntax rules. However, an SGML document is not necessarily an XML document.

XML was published by *World Wide Web Consortium* (W3C). Latest version of XML is 1.1 , and it is published in September, 2006.

1.2 Editors for writing XML document

There are many free editors by which an XML document could be developed. A few names are given as follows:

- Notepad ++

- XML Copy Editor
- Code Browser
- Microsoft XML Notepad
- XmlPad
- TextEdit

Initially, we shall take `notepad++` as an editor for writing XML documents. `notepad` also serves as an editor.

`notepad++` is a free editor and it supports several languages. Running in the MS Windows environment, its use is governed by GNU General Public License.

Based on the powerful editing component Scintilla, `notepad++` is written in `C++` and uses pure Win32 API and STL which ensures a higher execution speed and smaller program size.

At some point of time, we shall use *XML Copy Editor* to type XML documents. This is a free software. Using this software, we can validate DTD inserted in the document. Otherwise, a browser software ignores such validations. Initially, we use notepad++ editor for writing XML documents. At some point of time, we shall start writing XML documents using *XML Copy Editor* and validate the documents.

NOTE: GNU stands for *GNU's Not Unix*. The full form of API is Application Programming Interface. STL is a file format native to the stereolithography CAD software created by 3D Systems. STL has several acronyms such as *standard triangle language* and *standard tessellation language*.

1.3 Installing `notepad++` editor

It is a free software. Visit website https://notepad-plus-plus.org/downloads/ to download `notepad++` software. Otherwise, one can give a search using "download notepad++" using Google search. Follow the instructions to install the software and choose option to create a shortcut at the desktop.

1.4 Writing the first `XML` document

Every XML document has extension `.xml`. One can view an XML data in different ways.

▻ Firstly, one can just double click the icon of the file containing the XML code.

▻ One could write or copy the entire path of the file containing XML document in the address bar of a browser and then press the "Enter" key of the keyboard. Then the XML data gets displayed.

Here is an example of code file named as `company.xml`. Tags used in the XML document are user defined, where every tag `X`, there is a closing tag `</X>`.

Consider the XML document as given in Fig. 1.1. After execution, the output of the document is displayed on the screen, and the screenshot is given in Fig. 1.2. The title `company.xml` is displayed in the tab, placed above the screen. Unlike HTML document output, the file name of XML document and the title in the tab of the output screen are the same.

XML is a markup language, but not a programming language. It does not have features like variables, datatypes, operators, loops, functions, and conditional statements.

```
<academy>
<name>
Adhikari Academy of Learning
</name>
<subject>
Computer Science and Information Technology
</subject>
<address>
<location>
F-2, Kurtarkar Nagari 30, Shantinagar, Ponda
</location>
<state>
Goa
</state>
<pin>
403401
</pin>
<country>
India
</country>
</address>
</academy>
```

Figure 1.1: An XML document that displays details of the given company

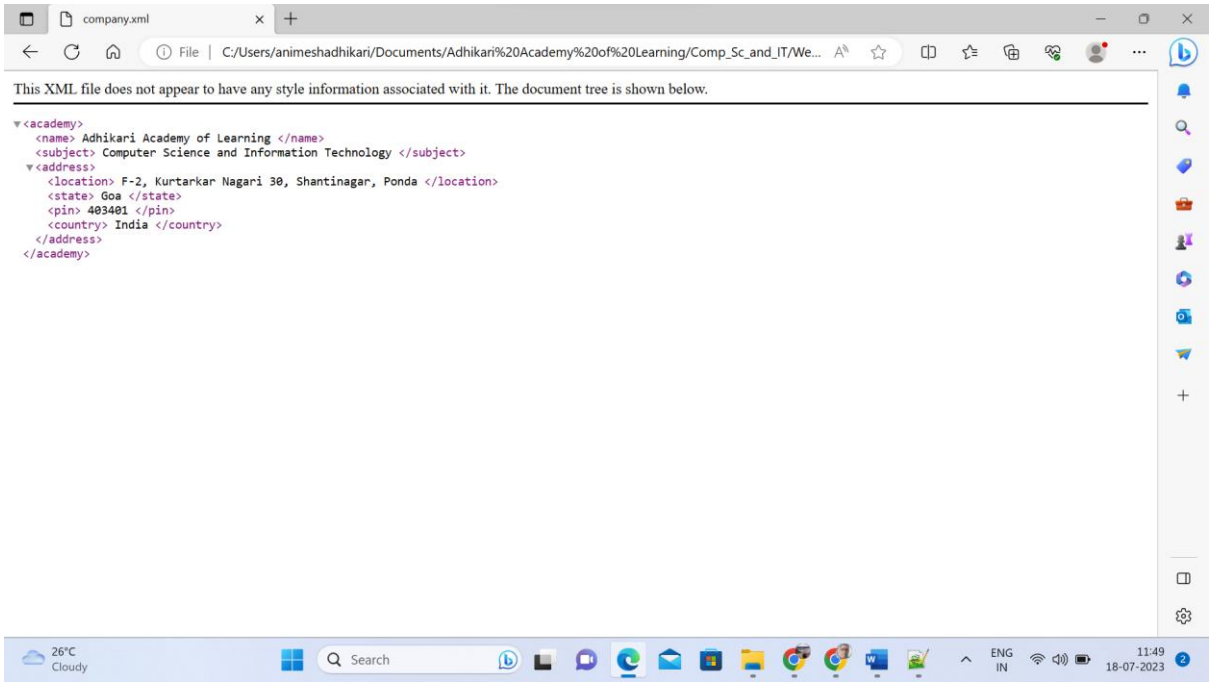

Figure 1.2: The output (screenshot) of the document `company.xml`

Note that there are two triangle shaped symbols in the output. Each triangle symbol corresponds to a user-defined tag. These symbols are clickable. Once a triangle symbol is clicked, it hides the content under this user-defined tag, and displays " . . . " in place of the content. In that sense, tags `<academy>` and `<address>` are collapsible.

Lecture 2

Contents: Parsing XML documents; Tags, text and elements; Rules for elements; Attributes and comments; Empty elements; XML declarations; PCDATA characters; Hierarchies in XML; Exercises

2.1 Parsing XML documents

XML parser is a software library or a package that provides interface for client applications to work with XML documents. It reads through the characters in the document, determines which characters are part of the document's markup and which are part of the document's data, and does all the other processing of an XML document that happens before an application can make use of the XML document. Modern browsers have built-in XML parsers. The goal of a parser is to transform XML into a readable code.

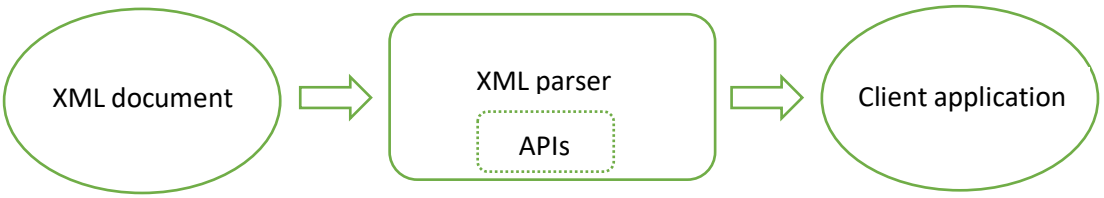

Figure 2.1: Block diagram of XML parsing

SAX (Simple API for XML) is an event-driven online algorithm for lexing and parsing XML documents. It is performant and memory efficient approach to parsing XML documents as it does not require the entire document be stored in memory.

DOM (Document Object Model) is another example of parser for XML documents. DOM approach uses a tree structure representation of an XML document instead of paginated events as with SAX.

Apache Xerces is an XML parser resulted from Apache Software Foundation's Xerces subproject. It is developed in Java and C++. This tool is free, and the distribution of the code is governed under the GNU Public License (GPL).

Expat is an XML parser toolkit written in C language. It is a free software. See http://expat.sourceforge.net for more details.

2.2 Tags, text and elements

Refer to XML document given in Fig. 1.1. The text starting with a < character and ending with a > character is an XML *tag*. Some examples of tag in the given document are `<academy>` and `<name>`. In fact, `<academy>` is an opening tag and `</academy>` is a closing tag. Tags are paired, so that any opening tag must have a closing tag. In XML parlance, these are called *start-tag* and *end-tag*. Another example of start-tag and end-tag are `<name>` and `</name>` respectively. The information in an XML document is contained within the various tags.

All of the information starting from the beginning of a start-tag to the end of an end-tag, and including everything in between, is called an *element*. An example is given below.

```
<subject> Computer Science and Information Technology </subject>
```

is an element, where

`<subject>` is a start-tag, and `</subject>` is an end-tag.

The text between the start-tag and end-tag of an element is called the *element content*. In this case, the element content is referred to as *parsed character data* (PCDATA), or with a more general term such as "text content" or "text node".

2.3 Rules for elements

We shall discuss some rules of elements in XML. These rules help in understanding XML documents.

- Every start-tag must have a matching end-tag. In HTML, such requirement is not mandatory. For example, one can start with `<p>` tag. But, it is not mandatory to use `</p>` to close it. In XML, end-tag is required, and it should match with the name of start-tag.
- Elements must be nested properly. One must close the child elements before closing their parents.
- An XML document can have only one root element. For example, XML document containing following lines is not well formed.

```
<name>Nikita Sharma</name>
<address>S-2, Kurtarkar Nagari, Shantinagar</address>
<town>Ponda</town>
<district>North Goa</district>
<state>Goa</state>
<pin>403401</pin>
```

  By adding a parent element `<person>`, the document becomes well formed. Well-formed document is given below.

```
<person>
   <name>Nikita Sharma</name>
   <address>S-2, Kurtarkar Nagari, Shantinagar</address>
   <town>Ponda</town>
   <district>North Goa</district>
   <state>Goa</state>
   <pin>403401</pin>
</person>
```

- Each element needs to obey naming conventions. Some XML naming conventions are given below.
  - A name can start with a letter or the dash (-) character, but not with a number or punctuation character.
  - A name cannot have space characters.
  - A name cannot contain the colon (:) character.
  - Names containing `xml`, in lowercase, uppercase or mixed case, are not valid.
- Tags in XML are case sensitive.
- Whitespaces for readability are allowed. Consider a small XML document as given above. There are space characters before `<name>`, `<address>`, `<town>`, `<district>`, `<state>`, and `<pin>` tags. These spaces are put for readability. Whitespace used for readability is called extraneous whitespace.

2.4 Attributes and comments

XML documents can also have attributes. Attributes come with name / value pair associated with an element. See below an XML document with attribute `gender` having value "Female".

```
<person gender = "Female">
   <name>Nikita Sharma</name>
   <address>S-2, Kurtarkar Nagari, Shantinagar</address>
    <town>Ponda</town>
    <district>North Goa</district>
    <state>Goa</state>
    <pin>403401</pin>
</person>
```

Comments are not really part of the document, but rather are intended for people who are reading the XML document. Using comments, one can annotate code, so that other people will be able to figure out what the code is doing. Comments start with the string `<!--` and end with the string `-->`, as shown below.

```
<person gender = "Female">
   <name>Nikita Sharma</name>
   <!-- near the ground -->
   <address>S-2, Kurtarkar Nagari, Shantinagar</address>
    <town>Ponda</town>
    <district>North Goa</district>
    <state>Goa</state>
    <pin>403401</pin>
</person>
```

A comment cannot be placed inside a tag. Also, it is not valid to put a double-dash string (`--`) inside a comment. Students are encouraged to run the document for checking errors by committing these mistakes.

2.5 Empty elements

An element that has no PCDATA is called an empty element. For example, `<milestone>` is an empty element in the document as given below.

```
<person gender = "Female">
   <name>Nikita Sharma</name>
   <!-- near the ground -->
   <address>S-2, Kurtarkar Nagari, Shantinagar</address>
   <milestone></milestone>
    <town>Ponda</town>
    <district>North Goa</district>
    <state>Goa</state>
    <pin>403401</pin>
</person>
```

Empty element `<milestone>` can be placed without end-tag also. We can replace the pair `<milestone></milestone>` by `<milestone/>`. In this case, `<milestone/>` is a *self-closing tag*.

2.6 XML declaration

XML provides declaration to label documents as being XML, along with giving the parsers some other information. A typical XML declaration is given as follows.

Lecture 3

Contents: Namespaces; How XML namespaces work; Default namespaces on descendants; Stylesheet for XML documents; Exercises

3.1 Namespaces

There may be some cases, where different document types often have elements with the same name, but with different meanings and semantics. XML uses namespaces to differentiate elements and attributes of different XML document types from each other when combining them into another document, or even when processing multiple documents at the same time. We shall create an XML document containing both information about a person and details of other persons in XHTML form. See Fig. 3.1. In this case, `<title>` in both places have different meanings. Unfortunately, XML parser does not differentiate these two `<title>` elements. These two <title> elements are placed with different colours to identify them quickly.

```
<?xml version="1.1"?>
 <person>
 <name>
 <title>Mr</title>
 <firstname>Rohit</firstname>
 <middle>Kumar</middle>
 <lastname>Bisth</lastname>
 <mobile>9444071781</mobile>
 <designation>Officer</designation>
 <department>Production</department>
 </name>
 <applicants>
 <html>
 <head><title>Details of job applicants</title></head>
 <body>
 <h1>Applicant:Kumkum Bera</h1>
 <p>Department:Sales</p>
 <p>ID:3241</p>
 <p>Mobile:8114071720</p>
 <h1>Applicant:Suhash Khan</h1>
 <p>Department:Information Technology</p>
 <p>ID:21</p>
 <p>Mobile:9774071820</p>
 </body>
 </html>
 </applicants>
 </person>
```

Figure 3.1: Two `<title>` elements from two documents as combined in an XML document

First `<title>` element provides salutation text of a person, whereas second one is the standard `<title>` element used in an XHTML document. The document (in Fig. 3.1) runs correctly, and the output is given in Fig. 3.2.

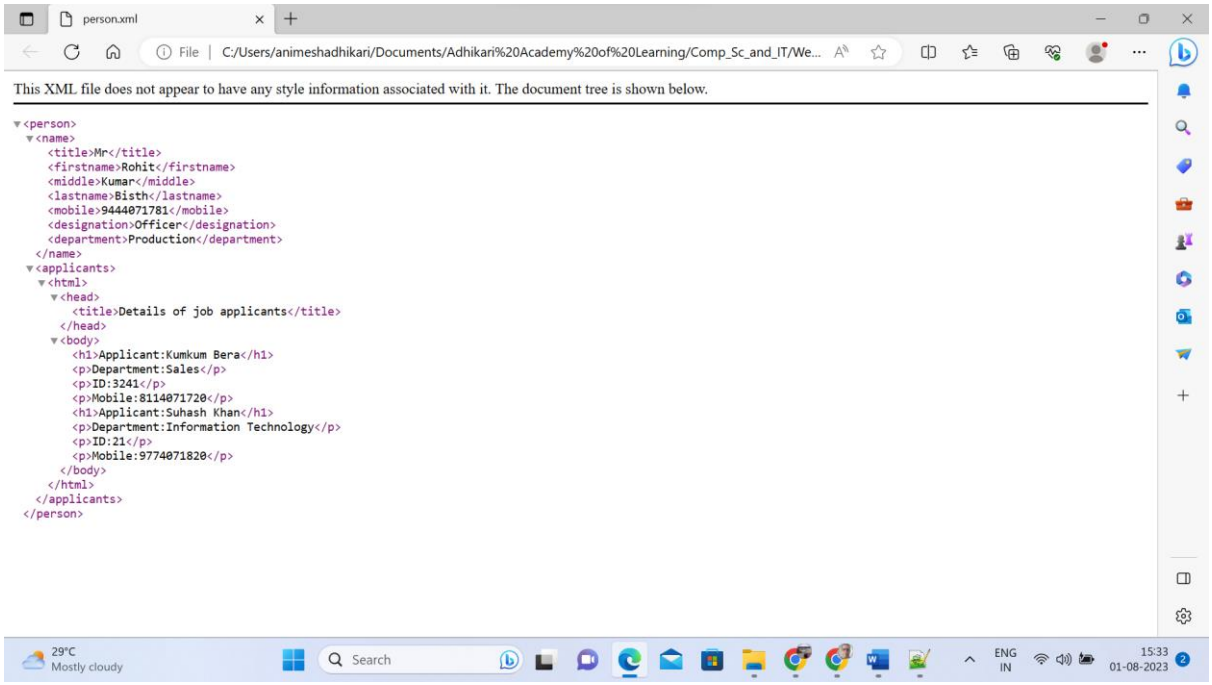

Figure 3.2: The output of document as given in Fig. 3.1

XML does not distinguish between the two `<title>` elements.

With respect to the given document, we have two categories of elements: `employee` and `xhtml`. We can prefix each element using either `employee` or `xhtml`. These categories are called *namespaces* as illustrated in Fig. 3.3.

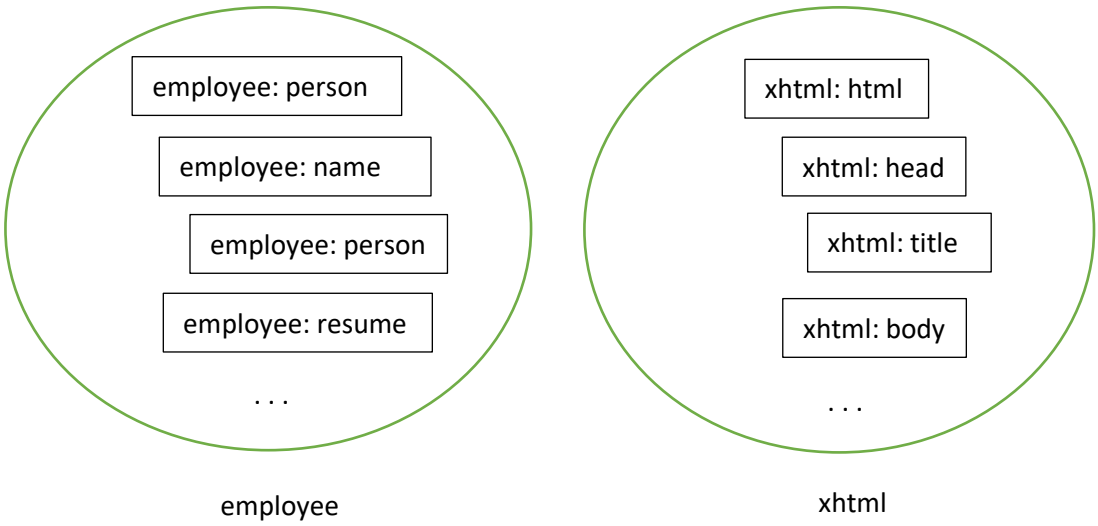

Figure 3.3: Two namespaces in XML document as given in Fig. 3.3

Using the namespace, earlier XML document is shown in Fig. 3.4.

```
<?xml version="1.0"?>
 <employee:person>
 <employee:name>
 <employee:title>Mr</employee:title>
 <employee:firstname>Rohit</employee:firstname>
 <employee:middle>Kumar</employee:middle>
 <employee:lastname>Bisth</employee:lastname>
 <employee:mobile>9444071781</employee:mobile>
 <employee:designation>Officer</employee:designation>
 <employee:department>Production</employee:department>
 </employee:name>
 <employee:applicants>
 <xhtml:html>
 <xhtml:head><xhtml:title>Details of job applicants</xhtml:title></xhtml:head>
 <xhtml:body>
 <xhtml:h1>Applicant:Kumkum Bera</xhtml:h1>
 <xhtml:p>Department:Sales</xhtml:p>
 <xhtml:p>ID:3241</xhtml:p>
 <xhtml:p>Mobile:8114071720</xhtml:p>
 <xhtml:h1>Applicant:Suhash Khan</xhtml:h1>
 <xhtml:p>Department:Information Technology</xhtml:d>
 <xhtml:p>ID:21</xhtml:p>
 <xhtml:p>Mobile:9774071820</xhtml:p>
 </xhtml:body>
 </xhtml:html>
 </employee:applicants>
 </employee:person>
```

Figure 3.3: XML document showing namespaces in association with elements

The document given in Fig. 3.3 does not execute. It gives many errors. Some errors are given below.

```
error on line 2 at column 18: Namespace prefix employee on person is
not defined
```

```
error on line 13 at column 13: Namespace prefix xhtml on html is not
defined
```

It appears that XML parser does not recognize namespaces `employee` and `xhtml` as given in Fig. 3.3.

NOTE: XHTML stands for *Extensible HyperText Markup Language*. It is almost identical to HTML but stricter than HTML, and a cross between HTML and XML languages. XHTML is an application of XML, a more restrictive subset of SGML. XHTML documents are well-formed and can be parsed using a standard XML parser.

3.2 How XML namespaces work

XML namespaces recommendation introduces a standard syntax for declaring namespaces and identifying the namespace by providing a URI for a given element or attribute in an XML document. For example, we use declaration for namespace `employee` as `<employee:person xmlns:employee="http://www.adhikariacademy.com/employee">`