Web Technologies

Unit 2: Contents

UNIT-II: Working with XML: Introduction, The syntax of XML, XML Document Structure, Document type Definition (DTD), Namespaces, XML schemas, XSLT, XML Parsers - DOM and SAX

Introduction to XML

1) What is XML?

A. XML stands for Extensible Markup Language
B. XML is a markup language much like HTML
C. XML was designed to carry data, not to display data
D. XML tags are not predefined. You must define your own tags
E. XML is designed to be self-descriptive
F. XML is a W3C Recommendation(World Wide Web Consortium)

XML Syntax

XML syntax refers to the rules that determine how an XML application can be written. The XML syntax is very straight forward, and this makes XML very easy to learn. Below are the main points to remember when creating XML documents.

XML documents must contain one **root** element that is the **parent** of all other elements

XML Document Structure

```
<? xml version="1.0" encoding="UTF-
  8"?>
<root>
    <child>
     <subchild>.....  </subchild>
    </child>
</root>
```

XML Prolog

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

is also known as **XML prolog**

Features of XML

Tag based language. tags are used to describe the content of the document

- Tags are defined by user and not by language
- Case sensitive and strict.
- Endorsed and maintained by **w3c** and used by all major companies like microsoft, oracle, Sun and IBM.
- Platform independent and language independent –any language on any platform can read process the data in xml document.
- With xml,white space is preserved

What is Well-formed XML

Ever xml document must be a well formed xml document. A well-formed XML is an document that compiles with following rules:

1. It has a single root element
2. Ever tag is opened and closed. Tags must be properly nested.
3. Values of an attribute is enclosed is quotes-single or double.

Course.xml

```
<? xml version=”1.0” encoding=“UTF-8” ?>
<course>
<name>java</name>
<fee  currency=”INR”>7500</fee>
<prereqsite> c  Language done by thomasa
   asfsafsafsafassfasfsafsafsafsafsafsa</prer
   eqsite>
</course>
```

Team.xml

```
<?xml version=”1.0” encoding=“UTF-8” ?>
  <team name=”Men InBule”
      <player>
          <name>M.S Dhoni</name>
           <age> 30</age>
       </player>
        <player>
            <name>ViratKohli</name>
             <age> 25</age>
         </player>
         <player>
              <name>Rohit</name>
               <age> 28</age>
         </player>
</team>
```

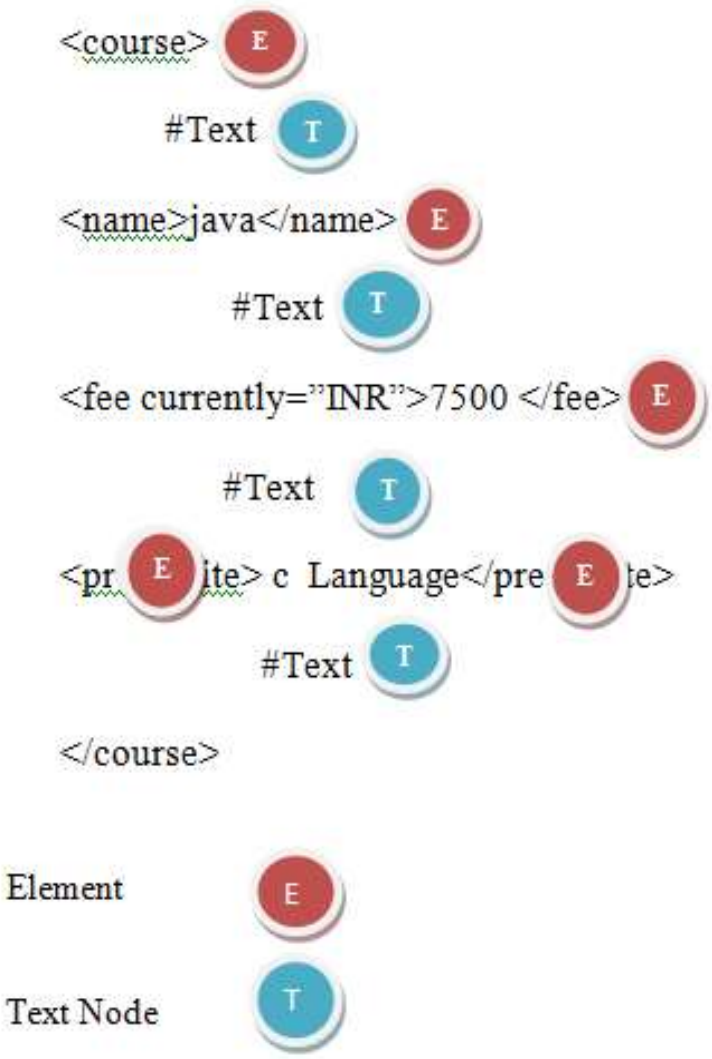
Why XML is Used?

We developers uses the XML files for following purposes:

- Storing the data for some application such as menu data or data for some comobox
- For developing the database driven websites.
- In image gallery application is can work as the data file (xml) /storing the names and location of the images.
- For shopping application it can be used to store the product details
- In travel applications XML data can be used to talk to booking gateway.
- On the web web services such as Weather services, Currency rates service etc. are using the XML language

Document Object Model

A DOM is a standard tree structure, where each node contains one of the components from an XML structure. The two most common types of nodes are **element nodes** and **text nodes**. Using DOM functions lets you create nodes, remove nodes, change their contents, and traverse the node hierarchy.



Why Use a DTD (Document Type Definition)

With a DTD, each of your XML files can carry a description of its own format. With a DTD, independent groups of people can agree to use a standard DTD for interchanging data. Your application can use a standard DTD to verify that the data you receive from the outside world is valid. You can also use a DTD to verify your own data. Seen from a DTD point of view, all XML documents (and HTML documents) are made up by the following building blocks:

1. Elements
2. Attributes
3. Entities
4. PCDATA (Parsed Character Data)
5. CDATA (Character Data)

PCDATA

- PCDATA means parsed character data.
- Think of character data as the text found between the start tag and the end tag of an XML element.
- PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and markup.
- Tags inside the text will be treated as markup and entities will be expanded.
- However, parsed character data should not contain any &, <, or > characters; these need to be represented by the & < and > entities, respectively.

CDATA

- CDATA means character data.
- CDATA is text that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will not be expanded.
- the qualifiers you can add to an element definition are listed in below:

Qualifier	Meaning
?	Optional(zero or more)
*	Zero or more
+	One or more

You can specify what type of data an element can contain parsed character data (PCDATA) or CDATA section ,which contain character data that is not parsed. The# that precedes PCDATA indicates that what follows is a special word rather than an element name.

Lab Task Exercise 4

Exercise 4:

Write an XML file which will display the Book information which includes the following: (i) Title of the book (ii) Author Name (iii) ISBN number (iv) Publisher name (v) Edition (vi) Price

(a) Write a Document Type Definition (DTD) to validate the above XML file.

(b) Write a XML Schema Definition (XSD)

Sample DTD for Book

```
<!ELEMENT bookdetails (book+)>
<!ELEMENT book (title,author,isbn,publisher,edition,price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

Book.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="book.xsl"?>
<!DOCTYPE book SYSTEM "book.dtd">
<bookdetails>
<book>
<title>XML Bible</title>
<author>Elliotte Rusty Harold</author>
<isbn>9876543210</isbn>
<publisher>Hungry Minds</publisher>
<edition>4th</edition>
<price>$21.99</price>
</book>
```

Contd..

```
<book>
<title>AI: A Modern Approach</title>
<author>Stuart J. Russell</author>
<isbn>9876543220</isbn>
<publisher>Princeton Hall</publisher>
<edition>6th</edition>
<price>$36.09</price>
</book>
<book>
<title>Beginning Java 2</title>
<author>Ivor Horton</author>
<isbn>9876543220</isbn>
<publisher>wrox</publisher>
<edition>3th</edition>
<price>$8.95</price>
</book>
```

Contd..

```
<book>
<title>HTML5: Up and Running</title>
<author>Mark Pilgrim</author>
<isbn>1234567890</isbn>
<publisher>O'REILLY</publisher>
<edition>1st</edition>
<price>$17.99</price>
</book>
</bookdetails>
```

book.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
  <body>
  <h2 style="color:green;" align="center">Books</h2>
  <table border="1" align="center">
    <tr style="color:grey;">
    <th>Title</th>
    <th>Author</th>
    <th>ISBN</th>
    <th>Publisher</th>
     <th>Edition</th>
     <th>Price</th>
     </tr>
```

Contd…

```
 <xsl:for-each select="bookdetails/book">
<tr style="width:70%">

<td style="font-family:'Comic Sans MS';
           color:red;width:70%">
    <xsl:value-of select="title"/>
</td>

<td style="text-transform: capitalize;
       font-weight:bold;" align="center">
    <xsl:value-of select="author"/>
</td>

<td style="color:blue">
     <xsl:value-of select="isbn"/>
</td>
```

```
  <td style="color:green; font-weight:bold;"
              align="center">
      <xsl:value-of select="publisher"/>
  </td>
   <td style="pink"  align="center">
      <xsl:value-of select="edition"/>
   </td>
    <td style="color:violet; font-weight:bold;">
      <xsl:value-of select="price"/>
   </td>
</tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Ways to declare DTD

- A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes.
- A DTD can be declared inline inside an XML document, or as an external reference.

Internal DTD Declaration

- If the DTD is declared inside the XML file, it should be wrapped in a DOCTYPE definition with the following syntax:

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

External DTD Declaration

- If the DTD is declared in an external file, it should be wrapped in a DOCTYPE definition with the following syntax:

<!DOCTYPE root-element SYSTEM "filename">

XML Namespaces

- XML Namespaces provide a method to avoid el

Name Conflicts

- In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

```
<table>
    <tr>
     <td>Apples</td>
     <td>Bananas</td>
    </tr>
</table>
```

- In the second XML file, the sample table element takes different values:

```
<table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
</table>
```

If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning. A user or an XML application will not know how to handle these differences.

Solving the Name Conflict Using a Prefix

Name conflicts in XML can easily be avoided using a name prefix.This XML carries information about an HTML table, and a piece of furniture:

```
<h:table>
    <h:tr>
     <h:td>Apples</h:td>
     <h:td>Bananas</h:td>
    </h:tr>
</h:table>
<f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
</f:table>
```

XML Namespaces - The xmlns Attribute

- When using prefixes in XML, a **namespace** for the prefix must be defined.
- The namespace can be defined by an **xmlns** attribute in the start tag of an element.
- The namespace declaration has the following syntax. xmlns:*prefix*="*URI*".

Here **Uniform Resource Identifier** (URI) is a string of characters which identifies an Internet Resource.

Example on XML Namespaces

```
<root xmlns:h="http://www.w3.org/TR/html4/"
    xmlns:f="https://www.w3schools.com/furniture">
<h:table>
      <h:tr>
       <h:td>Apples</h:td>
       <h:td>Bananas</h:td>
      </h:tr>
</h:table>
<f:table>
      <f:name>African Coffee Table</f:name>
      <f:width>80</f:width>
      <f:length>120</f:length>
</f:table>
</root>
```

XML schemas

- An XML Schema describes the structure of an XML document, just like a DTD.
- An XML document with correct syntax is called "Well Formed".
- An XML document validated against an XML Schema is both "Well Formed" and "Valid".
- XML Schemas are extensible to future additions
- XML Schemas are richer and more powerful than DTDs
- XML Schemas are written in XML
- XML Schemas Support data types
- XML Schemas supports namespace

Element	Meaning
<simpleType>	Describes a simple element
<complexType>	Describes a complex type
<element>	Describes a simple element
<choice>	Allows one of the specified element
<sequence>	Needs elements in the given sequence
<attribute>	Describes and attribute of an element

Most common datatypes

xs:string,

xs:decimal,

xs:integer,

xs:boolean,

xs:date,

xs:time

Example:

<xs:element name=”lastname” type=”xs:string”/>

<xs:element name=”age” type=”xs:integer”/>

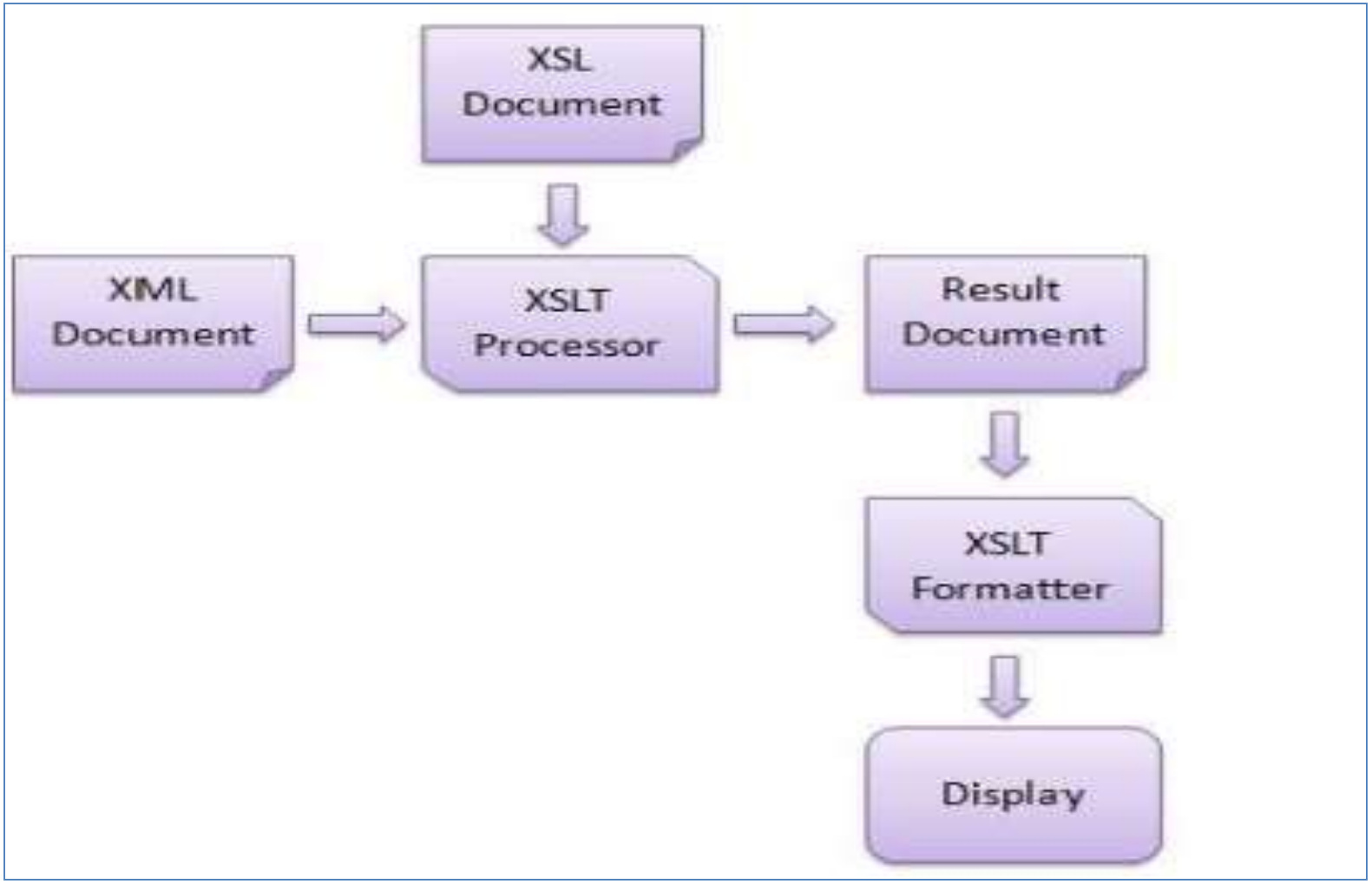
<xs:element name=”dateborn” type=”xs:date”/>

XSLT(Extensible Style Sheet Language Transformation)

Before XSLT, first we should learn about XSL. XSL stands for **eXtensible Style sheet Language**. It is a styling language for XML just like CSS is a styling language for HTML.

XSLT stands for **XSL Transformation**. It is used to transform XML documents into other formats (like transforming XML into HTML).

Architecture of XSLT



Syntax of XSLT

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

<xsl:stylesheet version="1.0“>

<xsl:template match="/">

<xsl:for-each select=“xml root element/parent element">

<xsl:value-of select=“xml child element name"/>

</xsl:for-each>

</xsl:template>

</xsl:stylesheet>

If we want to link xml style sheet for an XML page

<?xml-

stylesheet type = "text/xsl" href = "filename.xsl"**?>**

XML Parsers - DOM and SAX

There are two types of XML parsers namely Simple API for XML and Document Object Model.

1. SAX (Simple API for XML)
2. DOM(Document Object Model)

The objective of **DOM** (**Document Object Model**) parser and **SAX** (**Simple API for XML**) parser are same but implementation is different. Both the parser work in different way internally, but intent of both are same. Internal implementation of DOM Vs SAX is different. It means, with same intent philosophy of the implementation are different. In order to understand the difference between DOM and SAX, you have to understand each one of the parsers.

Key Differences between DOM & SAX parsers

- DOM parser load full XML file in-memory and creates a tree representation of XML document, while SAX is an event based XML parser and doesn't load whole XML document into memory.
- If you know you have sufficient amount of memory in your server you can choose DOM as this is faster because load entire XML in-memory and works as tree structure which is faster to access.
- As a thumb rule, for small and medium sized XML documents, DOM is much faster than SAX because of in memory management.
- As a thumb rule, for larger XML and for frequent parsing, SAX XML parser is better because it consume less memory.

Differences

	DOM (Document Object Model)	SAX (Simple API for XML) Parser
Abbreviation	**DOM** stands for Document Object Model,	**SAX** stands for Simple API for XML Parsing
type	Load entire memory and keep in tree structure	event based parse
size of Document	good for smaller size	good to choose for larger size of file.
Load	Load entire document in memory	does not load entire document.
suitable	better suitable for smaller and efficient memory	SAX is suitable for larger XML doc

THANK YOU