## XML MATERIAL

1. XML stands for e**X**tensible **M**arkup **L**anguage.
2. XML is a markup language much like HTML.
3. XML is designed to **transport** and **store data**.
4. XML was designed to **carry data**, not to **display data**.
5. XML tags are **not predefined**. You must define your own tags.
6. XML tags are **user defined tags**.
7. User can provide rules for the xml tags.
8. XML is **language neutral**.
9. XML is **operating system neutral**.
10. XML follows **tree structure**.
11. XML data can be shared among multiple applications.
12. XML is designed to be **self-descriptive**.
13. XML is a **W3C Recommendation**.
14. XML is **open Source**.
15. XML is supported by all **modern browsers**.
16. XML files must be saved with **.XML**
17. **XML is not a replacement for HTML**.
18. **XML and HTML were designed with different goals**.
19. XML was designed to **transport** and **store data**, with focus on **what data is**.
20. HTML was designed to **display data**, with focus on **how data looks**.

A **Well Formed XML** document is a document that conforms to the XML syntax rules like:

1. XML documents must have a **root element**.
2. XML elements must have a **opening tag**.
3. XML elements must have a **closing tag**.
4. XML tags are **case sensitive**.<email>….</EMAIL>
5. XML elements must be **properly nested**.
6. XML attribute values must always be **quoted**.

**Example(Demo.xml)**:

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

*<email>*

*<to>Ravi</to>*

*<from>Raju</from>*

*<subject>Reminder Message</subject>*

*<body>Don't forget meet this weekend</body>*

*</email>*

**DemoExample2:-**

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

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

<department>

<student branch="IT">

<regNo>16L31A1201</regNo>

<name>

<firstName>Ravi</firstName>

<middleName>Kumar</middleName>

<lastName>R.</lastName>

</name>

<dob>

<dd>20</dd>

<mm>2</mm>

<yyyy>1998</yyyy>

</dob>

<address>

Visakhapatnam

</address>

</student>

<student branch="IT">

<regNo>16L31A1202</regNo>

<name>

<firstName>Kiran</firstName>

<lastName>A</lastName>

</name>

<dob>

<dd>2</dd>

<mm>12</mm>

<yyyy>1998</yyyy>

</dob>

</student>

<student branch="IT">

<regNo>16L31A1203</regNo>

<name>

<firstName>Kiran</firstName>

<lastName>A</lastName>

</name>

<dob>

<dd>2</dd>

<mm>12</mm>

<yyyy>1998</yyyy>

</dob>

</student>

<student branch="CSE">

<regNo>16L31A0501</regNo>

<name>

<firstName>Kiran</firstName>

<lastName>A</lastName>

</name>

<dob>

<dd>21</dd>

<mm>12</mm>

<yyyy>1998</yyyy>

</dob>

</student>

<student branch="ECM">

<regNo>16L31A1901</regNo>

<name>

<firstName>Kiran</firstName>

<lastName>A</lastName>

</name>

<dob>

<dd>23</dd>

<mm>12</mm>

<yyyy>1998</yyyy>

</dob>

</student>

</department>

The following is a valid, well-formed XML file.

Using XML has the following advantages than using a binary or unstructured format:

* XML is plain text.
* XML represents data without defining how the data should be displayed.
* XML can be transformed into other formats via XSL.
* XML can be easily processed via standard parsers.
* XML files are hierarchical.

## Introduction to 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
* Internal DTD(declared inline inside an XML document)
* External DTD(an external reference).

### 1.Internal DTD Declaration:-

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]>

**Example XML document with an internal DTD(email.xml)**:

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

**<!DOCTYPE email [**

**<!ELEMENT email (to,from,subject,body)>**

**<!ELEMENT to (#PCDATA)>**

**<!ELEMENT from (#PCDATA)>**

**<!ELEMENT subject (#PCDATA)>**

**<!ELEMENT body (#PCDATA)>**

**]>**

**<email>**

**<to>Ravi</to>**

**<from>Raju</from>**

**<subject>Reminder Message</subject>**

**<body>Don't forget meet this weekend</body>**

**</email>**

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

**The DTD above is interpreted like this**:

* **!DOCTYPE email** defines that the root element of this document is note
* **!ELEMENT email** defines that the note element contains four elements: "to,from,subject,body"
* **!ELEMENT to** defines the to element to be of type "#PCDATA"
* **!ELEMENT from** defines the from element to be of type "#PCDATA"
* **!ELEMENT subject** defines the subject element to be of type "#PCDATA"
* **!ELEMENT body** defines the body element to be of type "#PCDATA

## 2.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">**

**Example(email.xml)**

<?xml version="1.0"?>

<!DOCTYPE email SYSTEM "email.dtd">

<email>

<to>Tove</to>

<from>Jani</from>

<subject type=’urgent’>Reminder</subject>

<body>Don't forget meet this weekend!</body>

</email>

|  |  |
| --- | --- |
| **Node Type** | **Example** |
| **Document type** | **<!DOCTYPE email SYSTEM "email.dtd">** |
| **Processing instruction** | **<?xml version="1.0"?>** |
| **Element** | **<subject type="urgent">Reminder</subject>** |
| **Attribute** | **type="urgent"** |
| **Text** | **Reminder** |

And this is the file "email.dtd" which contains the DTD:

<!ELEMENT email (to,from,heading,body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT subject (#PCDATA)>

<!ELEMENT body (#PCDATA)>

## 3.2 XML Schemas:

• XML Schema is an XML-based alternative to DTD.

• An XML schema describes the structure of an XML document.

• The XML Schema language is also referred to as XML Schema Definition (XSD). 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 namespaces.

• XML Schemas Support Data Types.

• XML Schema defines elements,attributes,elements having child elements,order of child elements,it also defines fixed and default values of elements and attibutes.

• The XML schema offers the facility of validating the xml document because the XML schema is used to describe the grammar and constraints on the elements of an XML instance document.

• Using the XML schema the user defined and built in data types can be used to express the constraints on the data more accurately.

• Reuse of data is possible using XML schema due to inheritance and groupings.

1. **XML support for data types**:

* It is easier to describe **allowable document content**.
* It is easier to validate the **correctness of data**.
* It is easier to work with **data from a database**.
* It is easier to define **data facets** (**restrictions on data**).
* It is easier to **define data patterns (data formats**).
* It is easier to **convert data between different data types**.

**2. XML Schemas use XML Syntax:-**

Another great strength about XML Schemas is that they are written in XML.Some benefits of that XML Schemas are written in XML:

* You don't have to learn a new language.
* You can use your XML editor to edit your Schema files.
* You can use your XML parser to parse your Schema files.
* You can manipulate your Schema with the XML DOM.
* You can transform your Schema with XSLT.

**3. XML Schemas are Extensible**

**4. XML Schemas Secure Data Communication**

The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD

**An XML Schema:**

* defines elements that can appear in a document
* defines attributes that can appear in a document
* defines which elements are child elements
* defines the order of child elements
* defines the number of child elements
* defines whether an element is empty or can include text
* defines data types for elements and attributes
* defines default and fixed values for elements and attributes

|  |  |  |
| --- | --- | --- |
| **No.** | **DTD** | **XSD** |
| 1. 1) | DTD stands for **Document Type Definition**. | XSD stands for XML Schema Definition. |
| 1. 2) | DTDs are derived from **SGML** syntax. | XSDs are written in XML. |
| 1. 3) | DTD **doesn't support datatypes**. | XSD **supports datatypes** for elements and attributes. |
| 1. 4) | DTD **doesn't support namespace**. | XSD **supports namespace**. |
| 1. 5) | DTD **doesn't define order** for child elements. | XSD **defines order** for child elements. |
| 1. 6) | DTD is **not extensible**. | XSD is **extensible**. |
| 1. 7) | DTD is **not easy to learn.**. | XSD is **easy to learn** because you don't need to learn new language.. |
| 1. 8) | DTD provides **less control** on XML structure. | XSD provides **more control** on XML structure. |

## 3.3 DOCUMENT OBJECT MODEL (DOM)

* The Document Object Model (DOM) is a **programming API** for HTML and XML documents.
* It defines the **logical structure** of documents and the way a **document is accessed and manipulated**.
* The Document Object Model can be used with any **programming language**.
* The DOM is **W3C Standard**.
* DOM is **Memory based**.
* The DOM represents a **tree view** of the XML document
* DOM is not **Language specific** or **platform specific**.
* According to the DOM, everything in an XML document is a **node**.
* The entire document is a document node, every XML element is an element node, every text in the xml is an element node, every attribute is attribute node, every comment is a comment node.
* A program called an **XML parser** can be used to load an XML document into the memory of your computer. When the document is loaded, it's information can be retrieved and manipulated by accessing the **Document Object Model (DOM)**.

(Books.xml)

<?xml version="1.0"?>

<bookstore>

<book category="cooking">

<title lang="en">Everyday Italian</title>

<author>Giada De</author>

<year>2005</year>

<price>300.00</price>

</book>

</bookstore>

**1.Program for (Read and print The book.xml file Contents)**

<html>

<body>

<script language="javascript">

var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")

xmlDoc.async="false"

xmlDoc.load("books.xml")

document.write("The first XML element in the file contains: ");

document.write("<br>"+xmlDoc.documentElement.childNodes.item(0).text);

document.write("<br>"+xmlDoc.documentElement.childNodes.item(1).text);

document.write("<br>"+xmlDoc.documentElement.childNodes.item(2).text);

document.write("<br>"+xmlDoc.documentElement.childNodes.item(3).text);

</script>

</body>

</html>

**2 . Program for (Reading and printing The Root Element of books.xml file)**

<html>

<body>

<script type="text/javascript">

var xmldoc=new ActiveXObject("Microsoft.XMLDOM");

xmldoc.load("books.xml");

document.writeln("<p>Here is the root node of the document:");

document.writeln("<br>"+xmldoc.documentElement.nodeName+"</br>");

</script>

</body>

</html>

**3. Program for (Reading and printing all the child node names of books.xml file)**

<html>

<body>

<script type="text/javascript">

var xmldoc=new ActiveXObject("Microsoft.XMLDOM");

xmldoc.load("books.xml");

document.writeln( "The following are its child elements" );

for (i = 0; i <xmldoc.documentElement.childNodes.length; i++) {

document.write("<br>"+xmldoc.documentElement.childNodes.item(i).nodeName);

}

</script>

</body>

</html>

**4. Program for (Reading and printing all the child node(title) values of books.xml file)**

<html>

<body>

<script type="text/javascript">

var xmldoc=new ActiveXObject("Microsoft.XMLDOM");

xmldoc.load("books.xml");

document.writeln( "The following are the title elements:<br>" );

x=xmldoc.getElementsByTagName('title');

for (i=0;i<x.length;i++)

{

document.write(x[i].childNodes[0].nodeValue);

}

</script>

</body>

</html>

**5. Program for (Reading and printing the title child node attribute values of books.xml file)**

<html>

<body>

<script type="text/javascript">

var xmldoc=new ActiveXObject("Microsoft.XMLDOM");

xmldoc.load("books.xml");

document.writeln( "<br>The following are the attribute elements:<br>" );

x=xmldoc.getElementsByTagName("book")[0].attributes;

document.write(x.getNamedItem("category").nodeValue);

</script>

</body>

</html>

**6. Program for (adding new node to books.xml file)**

<html>

<body>

<script type="text/javascript">

var xmldoc=new ActiveXObject("Microsoft.XMLDOM");

xmldoc.load("books.xml");

newNode=xmldoc.createElement("book");

x=xmldoc.documentElement;

y=xmldoc.getElementsByTagName("book");

document.write("Book elements before: " + y.length);

document.write("<br>");

x.insertBefore(newNode,y[3]);

y=xmldoc.getElementsByTagName("book");

document.write("Book elements after: " + y.length);

</script>

</body>

</html>

## 3.4 SAX(SIMPLE API for XML):-

Simple API for XML (SAX) parsing is different from DOM as it parses the XML files step by step and is event based model. The SAX parser triggers an event when they encounter an opening tag, element or attribute.

Unlike in DOM parser it is advisable to use the SAX parser for parsing large XML documents as it does not load the complete XML file in the memory. This parser parses node by node so it can read large XML files in smaller parts.

|  |  |
| --- | --- |
| DOM | SAX |
| Tree based parser (Tree of nodes) | Event based parser (Sequence of events) |
| DOM loads the file into the memory and then parse the file | SAX parses the file at it reads i.e. Parses node by node |
| Has memory constraints since it loads the whole XML file before parsing | No memory constraints as it does not store the XML content in the memory |
| DOM is read and write (can insert or delete the node) | SAX is read only i.e. can’t insert or delete the node |
| If the XML content is small then prefer DOM parser | Use SAX parser when memory content is large |
| Backward and forward search is possible. So this gives the ease of navigation | SAX reads the XML file from top to bottom and backward navigation is not possible |
| Slower at runtime | Faster at runtime |

### 3.5 XSLT

• XSLT stands for Extensible Style sheet Language for Transformations .

• XSLT allows to convert XML into other formats.

• XSL stands for Extensible Style sheet Language, and is a style sheet language for XML documents.

• XSLT - a language for transforming XML documents

• XPath - a language for navigating in XML documents

• XSL-FO - a language for formatting XML documents

• XSLT stands for XSL Transformations

• XSLT is the most important part of XSL

• XSLT transforms an XML document into another XML document

• XSLT uses XPath to navigate in XML documents

• XSLT is a W3C Recommendation

• All major browsers have support for XML and XSLT.

• XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.

• With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.

Example(books.xsl)Program for creating style sheet to books.xml:

(Save the below code with books.xsl)

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

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

<xsl:template match="/">

<html>

<body>

<h2>Book Collection</h2>

<table border="1">

<tr bgcolor="Green">

<th>Title</th>

<th>Author</th>

</tr>

<tr>

<td><xsl:value-of select="bookstore/book/title"/></td>

<td><xsl:value-of select="bookstore/book/author"/></td>

</tr>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

(Xml document adding books.xsl style sheet to Books.xml):-

(Save the Below code with books.xml)

<?xml version="1.0"?>

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

<bookstore>

<book category="programming">

<title lang="en">Java Programming</title>

<author>Herbeld Scheldt</author>

<year>2010</year>

<price>300.00</price>

</book>

</bookstore>

**2)Program for displaying the only those books whose price is greater than 500 from books1.xml in table format:**

(Save the below code with books1.xsl)

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

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

<html>

<body>

<h2>Book Collection</h2>

<table border="1">

<tr bgcolor="Green">

<th>Title</th>

<th>Author</th>

</tr>

<xsl:for-each select="bookstore/book">

<xsl:if test="price &gt;500">

<tr>

<td><xsl:value-of select="title"/></td>

<td><xsl:value-of select="author"/></td>

</tr>

</xsl:if>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

(Xml document adding booksl.xsl style sheet to Books.xm):-

(Save the Below code with books1.xml)

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="booksxsl1.xsl"?>

<bookstore>

<book category="programming">

<title lang="en">Java Programming</title>

<author>Herbeld Scheldt</author>

<year>2010</year>

<price>300.00</price>

</book>

<book category="designing">

<title lang="en">Adobe Flash</title>

<author>Hari</author>

<year>2013</year>

<price>2900</price>

</book>

<book category="web">

<title lang="en">JQuery</title>

<author>Ravindra</author>

<author>Praveen</author>

<author>Ranveer</author>

<author>Salman Khan</author>

<year>2011</year>

<price>550</price>

</book>

</bookstore>