**XML** - Introduction to XML, Advantages of XML, XML Tree, XML Attributes, XML DOM, DTD, XSD, XML with CSS, XSLT.

**1. Introduction to XML**

**1.1 Introduction**

XML stands for “extensible markup language”. XML was designed in 1996 and officially became a W3C standard in 1998. It was created to better represent data formats with a hierarchical structure.

Points need to be considered regarding XML,

* XML is designed to store and transport data but not to display data.
* XML is not a replacement for HTML.
* XML is designed to be self-descriptive.
* XML tags are not predefined. You must define your own tags.

**Example**

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

<contact-info>

<person>

<name> Vinoth Kannan R </name>

<company> K.L University </company>

<phone> 5698453278 </phone>

</person>

<person>

<name> Suresh Kumar B </name>

<company> K.L University </company>

<phone> 4376583986 </phone>

</person>

</contact-info>

**1.2 CSV vs XML Vs JSON**

The database which are similar to XML is CSV and JSON which are compared in the below table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Topic** | **CSV** | **XML** | **JSON** |
| **Abbreviation** | Comma Separated Values | Extensible Markup Language | Javascript Object Notation |
| **Data Format** | Eric,Andrea,Kusco | <person>  <name>  Eric  </name>  <age>  26  </age>  </person> | {“name”:”Eric”,”age”:”26″} |
| **Key Points** | Simplest structure reduces the bandwidth | Hierarchical data structures is appropriate when receiving complex data | Hierarchical data structures is appropriate when receiving complex data |

**1.3 XML vs Relational Database**

* XML data is hierarchical; relational data is represented in a model of logical relationships
* XML data is self-describing; relational data is not
* XML data has inherent ordering; relational data does not

**1.4 Factors influencing data model choice**

**When you need maximum flexibility:** Relational tables follow a fairly rigid model. For example, normalizing one table into many or denormalizing many tables into one can be very difficult. If the data design changes often, representing it as XML data is a better choice. XML schemas can be evolved over time, for example.

**When you need maximum performance for data retrieval:** Some expense is associated with serializing and interpreting XML data. If performance is more of an issue than flexibility, relational data might be the better choice.

**2. Advantages of XML**

* XML is platform independent and language independent.
* Most browsers have built in XML readers that allow you to inspect XML files.
* Since XML was the first standard hierarchical data format, most APIs have built in functionality to automatically convert XML data streams into native data structures like objects.

**3. XML Tree**

XML documents form a tree structure that starts at "the root" and branches to "the leaves".



**Fig. 3.1: XML Tree Structure**

## 3.1 XML Document

## <?xml version="1.0" encoding="UTF-8**"**?> <bookstore>    <book category="cooking">      <title lang="en">Everyday Italian</title>      <author>Giada De Laurentiis</author>      <year>2005</year>      <price>30.00</price>    </book>    <book category="children">      <title lang="en">Harry Potter</title>      <author>J K. Rowling</author>      <year>2005</year>      <price>29.99</price>    </book>    <book category="web">      <title lang="en">Learning XML</title>      <author>Erik T. Ray</author>      <year>2003</year>      <price>39.95</price>    </book> </bookstore>

# **4. XML Attributes**

XML elements can have attributes, just like HTML.

Attributes are designed to contain data related to a specific element.

**Example**

<person gender="male">

Some things to consider when using attributes are:

* attributes cannot contain multiple values (elements can)
* attributes cannot contain tree structures (elements can)
* attributes are not easily expandable (for future changes)

# **5. XML DOM**

The XML DOM defines a standard way for accessing and manipulating XML documents. It presents an XML document as a tree-structure.

This code retrieves the text value of the first <title> element in an XML document which is specified in XML Tree Section.

txt = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;

Output: txt = “Everyday Italian”

The complete example of xml storage and retrieval of data is given below.

**5.1 cd\_catlog.xml**

<CATALOG>

<CD>

<TITLE>Empire Burlesque</TITLE>

<ARTIST>Bob Dylan</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Columbia</COMPANY>

<PRICE>10.90</PRICE>

<YEAR>1985</YEAR>

</CD>

<CD>

<TITLE>Hide your heart</TITLE>

<ARTIST>Bonnie Tyler</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>CBS Records</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1988</YEAR>

</CD>

</CATLOG>

**5.2 Retrieving\_Data.html**

<!DOCTYPE html>

<html>

<body>

<p><button onclick="loadXMLDoc()">Get CD info</button></p>

<table id="demo" border="1">

<tr><th>Artist</th><th>Title</th></tr>

</table>

<script>

function loadXMLDoc() {

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

myFunction(this);

}

};

xmlhttp.open("GET", "cd\_catalog.xml" , true);

xmlhttp.send();

}

function myFunction(xml) {

var x, i, xmlDoc, table;

xmlDoc = xml.responseXML;

table = "<tr><th>Artist</th><th>Title</th></tr>";

x = xmlDoc.getElementsByTagName("CD")

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

table += "<tr><td>" +

x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +

"</td><td>" +

x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +

"</td></tr>";

}

document.getElementById("demo").innerHTML = table;

}

</script>

</body>

</html>

**5.3 Output**

Graphical user interface, application

Description automatically generated

**6. XML DTD**

A DTD is a Document Type Definition.

A DTD defines the structure and the legal elements and attributes of an XML document.

## Why Use a DTD?

With a DTD, independent groups of people can agree on a standard DTD for interchanging data.

An application can use a DTD to verify that XML data is valid.

## An Internal DTD Declaration

If the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> definition:

**6.1 Internal.xml**

<?xml version="1.0"?>  
<!DOCTYPE note [  
<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>  
]>  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend</body>  
</note>

## 6.2 An External DTD Declaration

If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file. Considering the DTD and XML file are in same location, the below example is been made.

**6.2.1 note.dtd**

<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>

**6.2.2 external.xml**

<?xml version="1.0"?>  
<!DOCTYPE note SYSTEM "note.dtd">  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>

## 6.3 Declaring Elements

In a DTD, XML elements are declared with the following syntax:

<!ELEMENT element-name category>  
or  
<!ELEMENT element-name (element-content)>

## 6.3.1 Empty Elements

Empty elements are declared with the category keyword EMPTY:

<!ELEMENT element-name EMPTY>  
**Example:**  
<!ELEMENT br EMPTY>  
**XML example:**  
<br />

## 6.3.2 Elements with Parsed Character Data

Elements with only parsed character data are declared with #PCDATA inside parentheses:

<!ELEMENT element-name (#PCDATA)>  
**Example:**  
<!ELEMENT from (#PCDATA)>

## 6.3.3 Elements with any Contents

Elements declared with the category keyword ANY, can contain any combination of parsable data:

<!ELEMENT element-name ANY>  
**Example:**  
<!ELEMENT note ANY>

## 6.3.4 Elements with Children (sequences)

Elements with one or more children are declared with the name of the children elements inside parentheses:

<!ELEMENT element-name (child1)>  
or  
<!ELEMENT element-name (child1,child2,...)>  
**Example:**  
<!ELEMENT note (to,from,heading,body)>

## 6.3.5 Declaring Only One Occurrence of an Element

<!ELEMENT element-name (child-name)>  
**Example:**  
<!ELEMENT note (message)>

The example above declares that the child element "message" must occur once, and only once inside the "note" element.

## 6.3.6 Declaring Minimum One Occurrence of an Element

<!ELEMENT element-name (child-name+)>  
**Example:**  
<!ELEMENT note (message+)>

The + sign in the example above declares that the child element "message" must occur one or more times inside the "note" element.

## 6.3.7 Declaring Zero or More Occurrences of an Element

<!ELEMENT element-name (child-name\*)>  
**Example:**  
<!ELEMENT note (message\*)>

The \* sign in the example above declares that the child element "message" can occur zero or more times inside the "note" element.

## 6.3.8 Declaring Zero or One Occurrences of an Element

<!ELEMENT element-name (child-name?)>  
**Example:**  
<!ELEMENT note (message?)>

The ? sign in the example above declares that the child element "message" can occur zero or one time inside the "note" element.

## 6.3.9 Declaring either/or Content

<!ELEMENT note (to,from,header,(message|body))>

The example above declares that the "note" element must contain a "to" element, a "from" element, a "header" element, and either a "message" or a "body" element.

## 6.3.10 Declaring Mixed Content

<!ELEMENT note (#PCDATA|to|from|header|message)\*>

The example above declares that the "note" element can contain zero or more occurrences of parsed character data, "to", "from", "header", or "message" elements.

## 6.4 Declaring Attributes

An attribute declaration has the following syntax:

<!ATTLIST element-name attribute-name attribute-type attribute-value>  
**DTD example:**  
<!ATTLIST payment type CDATA #REQUIRED  
**XML example:**  
<payment type="check" />

The attribute-type can be one of the following:

|  |  |
| --- | --- |
| **Type** | **Description** |
| CDATA | The value is character data |
| (*en1*|*en2*|..) | The value must be one from an enumerated list |
| ID | The value is a unique id |
| IDREF | The value is the id of another element |
| IDREFS | The value is a list of other ids |
| NMTOKEN | The value is a valid XML name |
| NMTOKENS | The value is a list of valid XML names |
| ENTITY | The value is an entity |
| ENTITIES | The value is a list of entities |
| NOTATION | The value is a name of a notation |
| xml: | The value is a predefined xml value |

The attribute-value can be one of the following:

|  |  |
| --- | --- |
| **Value** | **Explanation** |
| *Value* | The default value of the attribute |
| #REQUIRED | The attribute is required |
| #IMPLIED | The attribute is optional |
| #FIXED *value* | The attribute value is fixed |

**6.4. 1 Example**

**DTD:**  
<!ELEMENT square EMPTY>  
<!ATTLIST square width CDATA "0">  
**Valid XML:**  
<square width="100" />

In the example above, the "square" element is defined to be an empty element with a "width" attribute of  type CDATA. If no width is specified, it has a default value of 0.

# **7 XML Schema**

An XML Schema describes the structure of an XML document.

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

**7.1 Example**

**note.xsd**

<?xml version="1.0"?>  
<xs:schema xmlns:xs=<http://www.w3.org/2001/XMLSchema> targetNamespace=<https://www.w3schools.com> xmlns=<https://www.w3schools.com> elementFormDefault="qualified">

<xs:element name="note">  
<xs:complexType>  
<xs:sequence>  
 <xs:element name="to" type="xs:string"/>  
 <xs:element name="from" type="xs:string"/>  
 <xs:element name="heading" type="xs:string"/>  
 <xs:element name="body" type="xs:string"/>  
</xs:sequence>  
</xs:complexType>  
</xs:element>

</xs:schema>

**xml\_schema.xml**

<?xml version="1.0"?>  
<note xmlns=<https://www.w3schools.com> xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="https://www.w3schools.com/xml note.xsd">  
   <to>Tove</to>  
   <from>Jani</from>  
   <heading>Reminder</heading>  
   <body>Don't forget me this weekend!</body>  
</note>

**7.2 XSD Elements**

The syntax for defining a simple element is:

<xs:element name="xxx" type="yyy"/>

where xxx is the name of the element and yyy is the data type of the element.

XML Schema has a lot of built-in data types. The most common types are:

* xs:string
* xs:decimal
* xs:integer
* xs:boolean
* xs:date
* xs:time

Simple elements may have a default value OR a fixed value specified.

A default value is automatically assigned to the element when no other value is specified.

In the following example the default value is "red":

<xs:element name="color" type="xs:string" default="red"/>

# **7.3 XSD Attributes**

Simple elements cannot have attributes. If an element has attributes, it is considered to be of a complex type. But the attribute itself is always declared as a simple type.

The syntax for defining an attribute is:

<xs:attribute name="xxx" type="yyy"/>

where xxx is the name of the attribute and yyy specifies the data type of the attribute.

XML Schema has a lot of built-in data types. The most common types are:

* xs:string
* xs:decimal
* xs:integer
* xs:boolean
* xs:date
* xs:time

## 7.3.1 Default and Fixed Values for Attributes

Attributes may have a default value OR a fixed value specified.

A default value is automatically assigned to the attribute when no other value is specified.

In the following example the default value is "EN":

<xs:attribute name="lang" type="xs:string" default="EN"/>

A fixed value is also automatically assigned to the attribute, and you cannot specify another value.

In the following example the fixed value is "EN":

<xs:attribute name="lang" type="xs:string" fixed="EN"/>

**8. XML with CSS**

An XML file can be displayed using two ways. These are as follows :-

1. Cascading Style Sheet
2. Extensible Stylesheet Language Transformation

**8.1 Displaying XML file using CSS :**  
CSS can be used to display the contents of the XML document in a clear and precise manner. It gives the design and style to whole XML document.

**8.2 Basic steps in defining a CSS style sheet for XML :**  
For defining the style rules for the XML document, the following things shoulde be done :-

1. Define the style rules for the text elements such as font-size, color, font-weight, etc.
2. Define each element either as a block, inline or list element, using the display property of CSS.
3. Identify the titles and bold them.

**8.3 Linking XML with CSS :**  
In order to display the XML file using CSS, link XML file with CSS. Below is the syntax for linking the XML file with CSS:

**<?xml-stylesheet type="text/css" href="name\_of\_css\_file.css"?>**

**8.4 Example**  
In this example, the XML file is created that contains the information about five books and displaying the XML file using CSS.

**8.4.1 Books.xml**

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

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

<books>

    <heading>Welcome To GeeksforGeeks </heading>

    <book>

        <title>Title -: Web Programming</title>

        <author>Author -: Chrisbates</author>

        <publisher>Publisher -: Wiley</publisher>

        <edition>Edition -: 3</edition>

        <price> Price -: 300</price>

    </book>

    <book>

        <title>Title -: Internet world-wide-web</title>

        <author>Author -: Ditel</author>

        <publisher>Publisher -: Pearson</publisher>

        <edition>Edition -: 3</edition>

        <price>Price -: 400</price>

    </book>

    <book>

        <title>Title -: Computer Networks</title>

        <author>Author -: Foruouzan</author>

        <publisher>Publisher -: Mc Graw Hill</publisher>

        <edition>Edition -: 5</edition>

        <price>Price -: 700</price>

    </book>

    <book>

        <title>Title -: DBMS Concepts</title>

        <author>Author -: Navath</author>

        <publisher>Publisher -: Oxford</publisher>

        <edition>Edition -: 5</edition>

        <price>Price -: 600</price>

    </book>

    <book>

        <title>Title -: Linux Programming</title>

        <author>Author -: Subhitab Das</author>

        <publisher>Publisher -: Oxford</publisher>

        <edition>Edition -: 8</edition>

        <price>Price -: 300</price>

    </book>

</books>

**8.4.2 Rule.css**

books {

     color: white;

     background-color : gray;

     width: 100%;

}

 heading {

     color: green;

     font-size : 40px;

     background-color : powderblue;

}

 heading, title, author, publisher, edition, price {

     display : block;

}

 title {

     font-size : 25px;

     font-weight : bold;

}

**8.4.3 Output**

Graphical user interface, application

Description automatically generated

**9. XSLT**

XSLT stands for **E**xtensible **S**tylesheet **L**anguage **T**ransformation.

* XSLT is used to transform XML document from one form to another form.
* XSLT uses Xpath to perform matching of nodes to perform these transformation .
* The result of applying XSLT to XML document could be an another XML document, HTML, text or any another document from technology perspective.
* The XSL code is written within the XML document with the extension of *(.xsl)*.
* In other words, an XSLT document is a different kind of XML document.

**9.1 XML Namespace:** XML Namespaces are the unique names .

* XML Namespace is a mechanism by which element or attribute is assigned to a group.
* XML Namespace is used to avoid the name conflicts in the XML document.
* XML Namespace is recommended by W3C.

**XML Namespace Declaration:**  
It is declared using reserved attribute such as the attribute is *xmlns* or it can begin with *xmlns:*

**Syntax:**

<element xmlns:name = "URL">

where

* Namespace starts with the xmlns.
* The word name is the namespace prefix.
* the URL is the namespace identifier.

**9.2 Xpath:**

* Xpath is an important component of XSLT standard.
* Xpath is used to traverse the element and attributes of an XML document.
* Xpath uses different types of expression to retrieve relevant information from the XML document.
* Xpath contains a library of standard functions.

**9.3 Templates:**

* An XSL stylesheet contains one or more set of rules that are called templates.
* A template contains rules that are applied when the specific element is matched.
* An XSLT document has the following things:
  + The root element of the stylesheet.
  + A file of extension .xsl .
  + The syntax of XSLT i.e what is allowed and what is not allowed.
  + The standard namespace whose URL is [*http://www.w3.org/1999/XSL/Transform*](http://www.w3.org/1999/XSL/Transform).

**9.4 Students.xml**

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

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

 <student>

  <s>

   <name> Divyank Singh Sikarwar </name>

   <branch> CSE</branch>

   <age>18</age>

   <city> Agra </city>

  </s>

  <s>

   <name> Aniket Chauhan </name>

   <branch> CSE</branch>

   <age> 20</age>

   <city> Shahjahanpur </city>

  </s>

  <s>

   <name> Simran Agarwal</name>

   <branch> CSE</branch>

   <age> 23</age>

   <city> Buland Shar</city>

  </s>

  <s>

   <name> Abhay Chauhan</name>

   <branch> CSE</branch>

   <age> 17</age>

   <city> Shahjahanpur</city>

  </s>

  <s>

   <name> Himanshu Bhatia</name>

   <branch> IT</branch>

   <age> 25</age>

   <city> Indore</city>

  </s>

 </student>

**9.5 Rule.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>

  <h1 align="center">Students' Basic Details</h1>

   <table border="3" align="center" >

   <tr>

    <th>Name</th>

    <th>Branch</th>

    <th>Age</th>

    <th>City</th>

   </tr>

    <xsl:for-each select="student/s">

   <tr>

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

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

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

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

   </tr>

    </xsl:for-each>

    </table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

**9.6 Output**

Graphical user interface, application

Description automatically generated