Top of Form

XML (Extensible Markup Language) is a very popular simple text-based language that can be used as a mode of communication between different applications. It is considered as a standard means to transport and store data. JAVA provides excellent support and a rich set of libraries to parse, modify or inquire XML documents. This tutorial will teach you basic XML concepts and the usage of various types of Java based XML parsers in a simple and intuitive way.

#### What is XML?

XML is a simple text-based language which was designed to store and transport data in plain text format. It stands for Extensible Markup Language. Following are some of the salient features of XML.

* XML is a markup language.
* XML is a tag based language like HTML.
* XML tags are not predefined like HTML.
* You can define your own tags which is why it is called extensible language.
* XML tags are designed to be self-descriptive.
* XML is W3C Recommendation for data storage and data transfer.

#### Example

<?xml version = "1.0"?>

<Class>

<Name>First</Name>

<Sections>

<Section>

<Name>A</Name>

<Students>

<Student>Rohan</Student>

<Student>Mohan</Student>

<Student>Sohan</Student>

<Student>Lalit</Student>

<Student>Vinay</Student>

</Students>

</Section>

<Section>

<Name>B</Name>

<Students>

<Student>Robert</Student>

<Student>Julie</Student>

<Student>Kalie</Student>

<Student>Michael</Student>

</Students>

</Section>

</Sections>

</Class>

#### Advantages

Following are the advantages that XML provides −

* Technology agnostic − Being plain text, XML is technology independent. It can be used by any technology for data storage and data transfer purpose.
* Human readable − XML uses simple text format. It is human readable and understandable.
* Extensible − In XML, custom tags can be created and used very easily.
* Allow Validation − Using XSD, DTD and XML structures can be validated easily.

#### Disadvantages

Following are the disadvantages of using XML −

* Redundant Syntax − Normally XML files contain a lot of repetitive terms.
* Verbose − Being a verbose language, XML file size increases the transmission and storage costs.

#### XML Attributes

XML elements can have attributes. By the use of attributes we can add the information about the element.

XML attributes enhance the properties of the elements.

#### Note: XML attributes must always be quoted. We can use single or double quote.

Let us take an example of a book publisher. Here, book is the element and publisher is the attribute.

1. **<book** publisher="Tata McGraw Hill"**></book>**

Or

1. **<book** publisher='Tata McGraw Hill'**></book>**

**Metadata should be stored as attribute and data should be stored as element.**

1. **<book>**
2. **<book** category="computer"**>**
3. **<author>** A & B **</author>**
4. **</book>**

Data can be stored in attributes or in child elements. But there are some limitations in using attributes, over child elements.

#### Why should we avoid XML attributes

* Attributes cannot contain multiple values but child elements can have multiple values.
* Attributes cannot contain tree structure but child element can.
* Attributes are not easily expandable. If you want to change in attribute's vales in future, it may be complicated.
* Attributes cannot describe structure but child elements can.
* Attributes are more difficult to be manipulated by program code.
* Attributes values are not easy to test against a DTD, which is used to define the legal elements of an XML document.

#### Difference between attribute and sub-element

In the context of documents, attributes are part of markup, while sub elements are part of the basic document contents.

In the context of data representation, the difference is unclear and may be confusing.

Same information can be represented in two ways:

**1st way:**

1. **<book** publisher="Tata McGraw Hill"**>** **</book>**

**2nd way:**

1. **<book>**
2. **<publisher>** Tata McGraw Hill **</publisher>**
3. **</book>**

In the first example publisher is used as an attribute and in the second example publisher is an element.

Both examples provide the same information but it is good practice to avoid attribute in XML and use elements instead of attributes.

#### XML Tree Structure

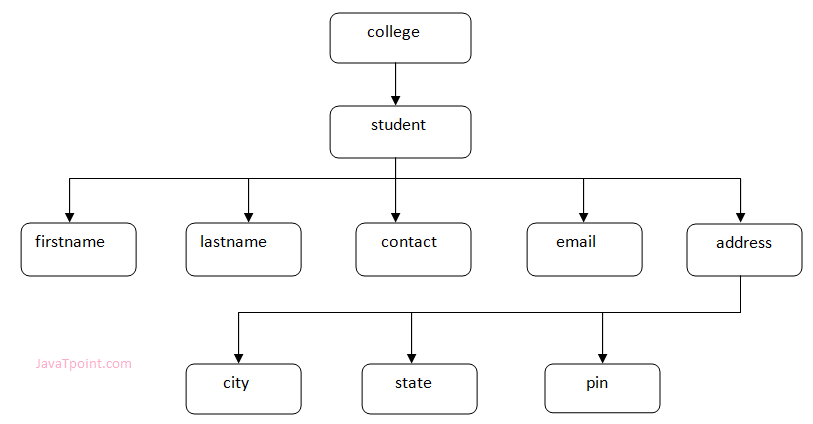
An XML document has a self descriptive structure. It forms a tree structure which is referred as an XML tree. The tree structure makes easy to describe an XML document.

A tree structure contains root element (as parent), child element and so on. It is very easy to traverse all succeeding branches and sub-branches and leaf nodes starting from the root.

#### Example of an XML document

1. **<?xml** version="1.0"**?>**
2. **<college>**
3. **<student>**
4. **<firstname>**Tamanna**</firstname>**
5. **<lastname>**Bhatia**</lastname>**
6. **<contact>**09990449935**</contact>**
7. **<email>**tammanabhatia@abc.com**</email>**
8. **<address>**
9. **<city>**Ghaziabad**</city>**
10. **<state>**Uttar Pradesh**</state>**
11. **<pin>**201007**</pin>**
12. **</address>**
13. **</student>**
14. **</college>**

Let's see the tree-structure representation of the above example.



In the above example, first line is the XML declaration. It defines the XML version 1.0. Next line shows the root element (college) of the document. Inside that there is one more element (student). Student element contains five branches named <firstname>, <lastname>, <contact>, <Email> and <address>.

<address> branch contains 3 sub-branches named <city>, <state> and <pin>.

#### Note: DOM parser represents the XML document in Tree structure.

#### XML Tree Rules

These rules are used to figure out the relationship of the elements. It shows if an element is a child or a parent of the other element.

**Descendants:** If element A is contained by element B, then A is known as descendant of B. In the above example "College" is the root element and all the other elements are the descendants of "College".

**Ancestors:** The containing element which contains other elements is called "Ancestor" of other element. In the above example Root element (College) is ancestor of all other elements.

Bottom of Form

Top of Form

#### XML Validation

A well formed XML document can be validated against DTD or Schema.

A well-formed XML document is an XML document with correct syntax. It is very necessary to know about valid XML document before knowing XML validation.

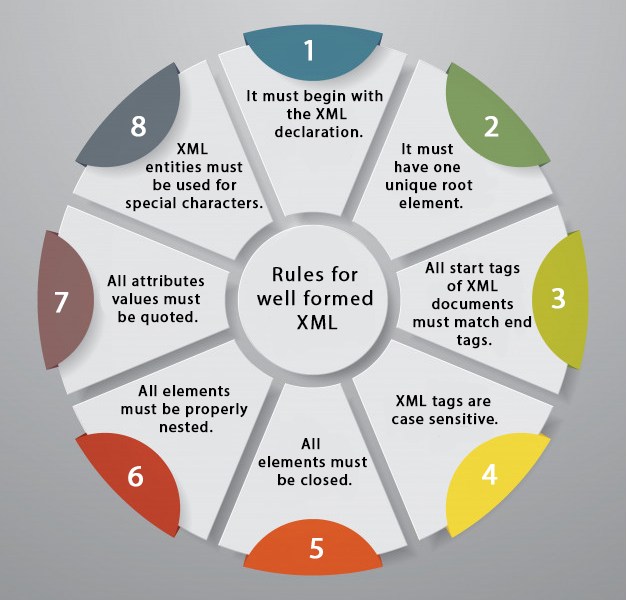
#### Valid XML document

It must be well formed (satisfy all the basic syntax condition)

It should be behave according to predefined DTD or XML schema

#### Rules for well formed XML

* It must begin with the XML declaration.
* It must have one unique root element.
* All start tags of XML documents must match end tags.
* XML tags are case sensitive.
* All elements must be closed.
* All elements must be properly nested.
* All attributes values must be quoted.
* XML entities must be used for special characters.



#### XML DTD

A DTD defines the legal elements of an XML document

In simple words we can say that a DTD defines the document structure with a list of legal elements and attributes.

XML schema is a XML based alternative to DTD.

Actually DTD and XML schema both are used to form a well formed XML document.

We should avoid errors in XML documents because they will stop the XML programs.

#### XML schema

It is defined as an XML language

Uses namespaces to allow for reuses of existing definitions

It supports a large number of built in data types and definition of derived data types

#### XML DTD

#### What is DTD

DTD stands for **Document Type Definition**. It defines the legal building blocks of an XML document. It is used to define document structure with a list of legal elements and attributes.

#### Purpose of DTD

Its main purpose is to define the structure of an XML document. It contains a list of legal elements and define the structure with the help of them.

#### Checking Validation

Before proceeding with XML DTD, you must check the validation. An XML document is called "well-formed" if it contains the correct syntax.

A well-formed and valid XML document is one which have been validated against DTD.

Visit **http://www.xmlvalidation.com** to validate the XML file.

#### Valid and well-formed XML document with DTD

Let's take an example of well-formed and valid XML document. It follows all the rules of DTD.

employee.xml

1. **<?xml** version="1.0"**?>**
2. <!DOCTYPE employee SYSTEM "employee.dtd"**>**
3. **<employee>**
4. **<firstname>**vimal**</firstname>**
5. **<lastname>**jaiswal**</lastname>**
6. **<email>**vimal@javatpoint.com**</email>**
7. **</employee>**

In the above example, the DOCTYPE declaration refers to an external DTD file. The content of the file is shown in below paragraph.

employee.dtd

1. <!ELEMENT employee (firstname,lastname,email)**>**
2. <!ELEMENT firstname (#PCDATA)**>**
3. <!ELEMENT lastname (#PCDATA)**>**
4. <!ELEMENT email (#PCDATA)**>**

#### Description of DTD

**<!DOCTYPE employee :** It defines that the root element of the document is employee.

**<!ELEMENT employee:** It defines that the employee element contains 3 elements "firstname, lastname and email".

**<!ELEMENT firstname:** It defines that the firstname element is #PCDATA typed. (parse-able data type).

**<!ELEMENT lastname:** It defines that the lastname element is #PCDATA typed. (parse-able data type).

**<!ELEMENT email:** It defines that the email element is #PCDATA typed. (parse-able data type).

#### XML DTD with entity declaration

A doctype declaration can also define special strings that can be used in the XML file.

An entity has three parts:

1. An ampersand (&)
2. An entity name
3. A semicolon (;)

Syntax to declare entity:

1. <!ENTITY entity-name "entity-value"**>**

Let's see a code to define the ENTITY in doctype declaration.

author.xml

1. **<?xml** version="1.0" standalone="yes" **?>**
2. <!DOCTYPE author [
3. <!ELEMENT author (#PCDATA)**>**
4. <!ENTITY sj "Sonoo Jaiswal"**>**
5. ]**>**
6. **<author>**&sj;**</author>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/author.xml)In the above example, sj is an entity that is used inside the author element. In such case, it will print the value of sj entity that is "Sonoo Jaiswal".

#### Note: A single DTD can be used in many XML files.

#### CDATA vs PCDATA

#### CDATA

CDATA: (Unparsed Character data): CDATA contains the text which is not parsed further in an XML document. Tags inside the CDATA text are not treated as markup and entities will not be expanded.

Let's take an example for CDATA:

1. **<?xml** version="1.0"**?>**
2. <!DOCTYPE employee SYSTEM "employee.dtd"**>**
3. **<employee>**
4. <![CDATA[
5. <firstname>vimal</firstname>
6. <lastname>jaiswal</lastname>
7. <email>vimal@javatpoint.com</email>
8. ]]>
9. **</employee>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/cdata.xml)In the above CDATA example, CDATA is used just after the element employee to make the data/text unparsed, so it will give the value of employee:

<firstname>vimal</firstname><lastname>jaiswal</lastname><email>vimal@javatpoint.com</email>

#### PCDATA

PCDATA: (Parsed Character Data): XML parsers are used to parse all the text in an XML document. PCDATA stands for Parsed Character data. PCDATA is the text that will be parsed by a parser. Tags inside the PCDATA will be treated as markup and entities will be expanded.

In other words you can say that a parsed character data means the XML parser examine the data and ensure that it doesn't content entity if it contains that will be replaced.

Let's take an example:

1. **<?xml** version="1.0"**?>**
2. <!DOCTYPE employee SYSTEM "employee.dtd"**>**
3. **<employee>**
4. **<firstname>**vimal**</firstname>**
5. **<lastname>**jaiswal**</lastname>**
6. **<email>**vimal@javatpoint.com**</email>**
7. **</employee>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/pcdata.xml)In the above example, the employee element contains 3 more elements 'firstname', 'lastname', and 'email', so it parses further to get the data/text of firstname, lastname and email to give the value of employee as:

vimal jaiswal vimal@javatpoint.com

Bottom of Form

Top of Form

**Video Lectures :**

**Reference Links:**

<https://www.javatpoint.com/xml-tutorial>

<https://www.javatpoint.com/xml-example>

<https://www.infoworld.com/article/2076282/programming-xml-in-java--part-1.html>

https://www.journaldev.com/1240/java-xml-parser

Bottom of Form