1. JAVA XML – OVERVIEW

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.

2. JAVA XML – PARSERS

XML Parsing refers to going through an XML document in order to access or modify data.

What is XML Parser?

XML Parser provides a way to access or modify data in an XML document. Java provides multiple options to parse XML documents. Following are the various types of parsers which are commonly used to parse XML documents.

- **Dom Parser** - Parses an XML document by loading the complete contents of the document and creating its complete hierarchical tree in memory.
- **SAX Parser** - Parses an XML document on event-based triggers. Does not load the complete document into the memory.
- **JDOM Parser** - Parses an XML document in a similar fashion to DOM parser but in an easier way.
- **StAX Parser** - Parses an XML document in a similar fashion to SAX parser but in a more efficient way.
- **XPath Parser** - Parses an XML document based on expression and is used extensively in conjunction with XSLT.
- **DOM4J Parser** - A java library to parse XML, XPath, and XSLT using Java Collections Framework. It provides support for DOM, SAX, and JAXP.

There are JAXB and XSLT APIs available to handle XML parsing in object-oriented way. We'll elaborate each parser in detail in the subsequent chapters of this tutorial.

3. JAVA DOM PARSER – OVERVIEW

The Document Object Model (DOM) is an official recommendation of the World Wide Web Consortium (W3C). It defines an interface that enables programs to access and update the style, structure, and contents of XML documents. XML parsers that support DOM implement this interface.

When to Use?

You should use a DOM parser when:

- You need to know a lot about the structure of a document.
- You need to move parts of an XML document around (you might want to sort certain elements, for example).
- You need to use the information in an XML document more than once.

What you get?

When you parse an XML document with a DOM parser, you get back a tree structure that contains all of the elements of your document. The DOM provides a variety of functions you can use to examine the contents and structure of the document.

Advantages

The DOM is a common interface for manipulating document structures. One of its design goals is that Java code written for one DOM-compliant parser should run on any other DOM-compliant parser without having to do any modifications.

DOM Interfaces

The DOM defines several Java interfaces. Here are the most common interfaces:

- **Node** - The base datatype of the DOM.
- **Element** - The vast majority of the objects you'll deal with are Elements.
- **Attr** - Represents an attribute of an element.
- **Text** - The actual content of an Element or Attr.
- **Document** - Represents the entire XML document. A Document object is often referred to as a DOM tree.

Common DOM Methods

When you are working with DOM, there are several methods you'll use often:

- **Document.getDocumentElement()** - Returns the root element of the document.
- **Node.getFirstChild()** - Returns the first child of a given Node.
- **Node.getLastChild()** - Returns the last child of a given Node.
- **Node.getNextSibling()** - These methods return the next sibling of a given Node.
- **Node.getPreviousSibling()** - These methods return the previous sibling of a given Node.
- **Node.getAttribute(attrName)** - For a given Node, it returns the attribute with the requested name.

```
getChildNodes();
```

Demo Example

Here is the input XML file that we need to parse:

```
<?xml version="1.0"?>
<class>
   <student rollno="393">
      <firstname>dinkar</firstname>
      <lastname>kad</lastname>
      <nickname>dinkar</nickname>
      <marks>85</marks>
   </student>
   <student rollno="493">
      <firstname>Vaneet</firstname>
      <lastname>Gupta</lastname>
      <nickname>vinni</nickname>
```

```
      <marks>95</marks>
   </student>
   <student rollno="593">
      <firstname>jasvir</firstname>
      <lastname>singn</lastname>
      <nickname>jazz</nickname>
      <marks>90</marks>
   </student>
</class>
```

DomParserDemo.java

```
package com.tutorialspoint.xml;

import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;

public class DomParserDemo {
   public static void main(String[] args){

      try {
         File inputFile = new File("input.txt");
         DocumentBuilderFactory dbFactory
            = DocumentBuilderFactory.newInstance();
         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
         Document doc = dBuilder.parse(inputFile);
         doc.getDocumentElement().normalize();
         System.out.println("Root element :"
```

```
    + doc.getDocumentElement().getNodeName());
NodeList nList = doc.getElementsByTagName("student");
System.out.println("----------------------------");
for (int temp = 0; temp < nList.getLength(); temp++) {
    Node nNode = nList.item(temp);
    System.out.println("\nCurrent Element :"
        + nNode.getNodeName());
    if (nNode.getNodeType() == Node.ELEMENT_NODE) {
        Element eElement = (Element) nNode;
        System.out.println("Student roll no : "
            + eElement.getAttribute("rollno"));
        System.out.println("First Name : "
            + eElement
            .getElementsByTagName("firstname")
            .item(0)
            .getTextContent());
        System.out.println("Last Name : "
        + eElement
            .getElementsByTagName("lastname")
            .item(0)
            .getTextContent());
        System.out.println("Nick Name : "
        + eElement
            .getElementsByTagName("nickname")
            .item(0)
            .getTextContent());
        System.out.println("Marks : "
        + eElement
            .getElementsByTagName("marks")
            .item(0)
            .getTextContent());
    }
```

```
            }
         } catch (Exception e) {
            e.printStackTrace();
         }
      }
   }
```

This would produce the following result:

```
Root element :class
----------------------------

Current Element :student
Student roll no : 393
First Name : dinkar
Last Name : kad
Nick Name : dinkar
Marks : 85

Current Element :student
Student roll no : 493
First Name : Vaneet
Last Name : Gupta
Nick Name : vinni
Marks : 95

Current Element :student
Student roll no : 593
```

```
First Name : jasvir
Last Name : singn
Nick Name : jazz
Marks : 90
```