# Java JDOM Parser - Parse XML Document

Java JDOM Parser is an open source API in Java that has classes and methods to parse XML documents. JDOM provides random access of XML elements as it creates a tree document structure inside the memory using DOMBuilder or SAXBuilder. In this chapter, we are going to see how to build a JDOM document from an XML file using a SAX Parser.

## Parse XML Using JDOM Parser

Following are the steps used while parsing a document using JDOM Parser –

- **Step 1:** Creating a SAXBuilder Object
- **Step 2:** Reading the XML
- **Step 3:** Parsing the XML Document
- **Step 4:** Retrieving the Elements

## Step 1: Creating a SAXBuilder Object

JDOM document is build using a SAX Parser as follows –

```
SAXBuilder saxBuilder = new SAXBuilder();
```

We can also create JDOM document using an already existing DOM org.w3c.dom.Document as follows –

```
DOMBuilder domBuilder = new DOMBuilder();
```

## Step 2: Reading the XML

An XML file is taken into a File object as follows –

```
File xmlFile = new File("input.xml");
```

We can also take XML content using StringBuilder object. Later, we can convert it into bytes for parsing.

```
StringBuilder xmlBuilder = new StringBuilder();
xmlBuilder.append("<?xml version="1.0"?> <rootElement></rootElement>");
```

```
ByteArrayInputStream input = new ByteArrayInputStream(
xmlBuilder.toString().getBytes("UTF-8"));
```

## Step 3: Parsing the XML Document

Using build() function, we parse an XML file or input stream. It builds the JDOM document from the given file or input stream. It throws JDOMException and IOException when there are errors in parsing the document.

```
Document document = saxBuilder.build(input);
```

## Step 4: Retrieving the Elements

After following the first three steps, we have successfully build JDOM document from our XML file or stream. We can now use methods available in Document and Element classes to obtain all the related information from the document.

## Retrieving Root Element

The method **getRootElement()** of Document interface returns the root element of the document in the form of an Element object.

The **getName()** method on Element object returns the name of the element in the form of a String.

## Example

The following **RetrieveRootElement.java** program takes XML content in a StringBuilder object. It is then converted into bytes and parsed using build() function. It retrieves the root element and prints the name of the root element.

```
import java.io.ByteArrayInputStream;
import org.jdom2.Document;
import org.jdom2.Element;
import org.jdom2.input.SAXBuilder;

public class RetrieveRootElement {
   public static void main(String args[]) {
      try {
         //Creating a SAXBuilder Object
         SAXBuilder saxBuilder = new SAXBuilder();
```

```
         //Reading the XML
         StringBuilder xmlBuilder = new StringBuilder();
         xmlBuilder.append("<class></class>");
         ByteArrayInputStream input = new
ByteArrayInputStream(xmlBuilder.toString().getBytes("UTF-8"));

         //Parsing the XML Document
         Document document = saxBuilder.build(input);

         //Retrieving the Root Element Name
         Element root_element = document.getRootElement();
         System.out.println("Root Element Name : " + root_element.getName());

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

## Output

The root element name, "class" is printed on the output screen.

```
Root Element Name : class
```

Learn **Java** in-depth with real-world projects through our **Java certification course**. Enroll and become a certified expert to boost your career.

# Retrieving Child Elements

To retrieve child elements of an element, **getChildren()** method is used on the Element object. It returns the child elements in the form of a list. This list contains all the child elements in the of Element objects.

To retrieve text content of an element, **getText()** method is used on the Element object. It returns the content between the opening and closing tags of an Element.

## Example

Let us add three student child elements to our class element and save this file as **student.xml**. The name of the student is mentioned in the text content of each student element.

```
<?xml version = "1.0"?>
<class>
   <student>dinkar</student>
   <student>Vaneet</student>
   <student>jasvir</student>
</class>
```

Now, the following java program reads the student.xml file and retrieves all the child elements along with their text content.

```
import java.io.File;
import java.util.List;
import org.jdom2.Document;
import org.jdom2.Element;
import org.jdom2.input.SAXBuilder;

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

         //Creating a SAXBuilder Object
         SAXBuilder saxBuilder = new SAXBuilder();

         //Reading the XML
         File inputFile = new File("student.xml");

         //Parsing the XML Document
         Document document = saxBuilder.build(inputFile);

         //Retrieving Root Element
         Element RootElement = document.getRootElement();
         System.out.println("Root element :" + RootElement.getName());

         //Retrieving Child Elements
         List<Element> studentList = RootElement.getChildren();
         System.out.println("----------------------------");

         for (int temp = 0; temp < studentList.size(); temp++) {
            Element student = studentList.get(temp);
            System.out.println("\nCurrent Element :" + student.getName());
            System.out.println("Text Content :" + student.getText());
```

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

## Output

All the three child elements are displayed with their text content.

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

Current Element :student
Text Content :dinkar

Current Element :student
Text Content :Vaneet

Current Element :student
Text Content :jasvir
```

# Retrieving Attributes

The **getAttribute("attr_name")** method on an Element object takes attribute name as an argument and retrieves the attribute in the form of Attribute object. If there is no such attribute in an element, it returns null.

The **getValue()** method on an Attribute object retrieves the value of the attribute as textual content.

## Example

To **student.xml** file, let us add some child elements to student element along with the attribute, "rollno". Now, let us try to retrieve all this information using JDOM parser API.

```
<?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>
```

In the following **RetrievingAttributes.java** program, we have first collected all the child elements in an Element list and then used getChild() method to get the details of each child inside the student element.

```
import java.io.File;
import java.util.List;
import org.jdom2.Attribute;
import org.jdom2.Document;
import org.jdom2.Element;
import org.jdom2.input.SAXBuilder;


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

         //Creating a SAXBuilder Object
         SAXBuilder saxBuilder = new SAXBuilder();

         //Reading the XML
         File inputFile = new File("student.xml");

         //Parsing the XML Document
```

```
         Document document = saxBuilder.build(inputFile);

         //Retrieving Root Element
         Element RootElement = document.getRootElement();
         System.out.println("Root element :" + RootElement.getName());

         //Retrieving Child Elements and Attributes
         List<Element> studentList = RootElement.getChildren();
         System.out.println("----------------------------");

         for (int temp = 0; temp < studentList.size(); temp++) {
            Element student = studentList.get(temp);
            System.out.println("\nCurrent Element :"
               + student.getName());
            Attribute attribute =  student.getAttribute("rollno");
            System.out.println("Student roll no : "
               + attribute.getValue() );
            System.out.println("First Name : "
               + student.getChild("firstname").getText());
            System.out.println("Last Name : "
               + student.getChild("lastname").getText());
            System.out.println("Nick Name : "
               + student.getChild("nickname").getText());
            System.out.println("Marks : "
               + student.getChild("marks").getText());
         }
      } catch(Exception e) {
         e.printStackTrace();
      }
   }
}
```

## Output

Information of each student is dispalyed along with their roll numbers.

```
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
```