Java JDOM Parser - Overview

JDOM is an open source, java based library to parse XML document and it is typically java developer friendly API. It is java optimized, it uses java collection like List and Arrays. It works with DOM and SAX APIs and combines the best of the two. It is of low memory footprint and is nearly as fast as SAX.

## Environment Setup

In order to use JDOM parser, you should have jdom.jar in your application's classpath. Download [**jdom-2.0.5.zip.**](http://www.jdom.org/dist/binary/jdom-2.0.5.zip)

## When to use?

You should use a JDOM parser when:

* You need to know a lot about the structure of a document
* You need to move parts of the document around (you might want to sort certain elements, for example)
* You need to use the information in the document more than once
* You are a java developer and want to leverage java optimized parsing of XML.

## What you get?

When you parse an XML document with a JDOM parser, you get the flexibility to get back a tree structure that contains all of the elements of your document without impacting the memory footprint of the application. The JDOM provides a variety of utility functions you can use to examine the contents and structure of the document in case document is well structured and its structure is known.

## Advantages

JDOM gives java developers flexibility and easy maintainablity of xml parsing code. It is light weight and quick API.

## JDOM classes

The JDOM defines several Java classes. Here are the most common classes:

* **Document** - Represents the entire XML document. A Document object is often referred to as a DOM tree.
* **Element** - Represents an XML element. Element object has methods to manipulate its child elements,its text, attributes and namespaces.
* **Attribute** Represents an attribute of an element. Attribute has method to get and set the value of attribute. It has parent and attribute type.
* **Text** Represents the text of XML tag.
* **Comment** Represents the comments in a XML document.

## Common JDOM methods

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

* **SAXBuilder.build(xmlSource)()** - Build the JDOM document from the xml source.
* **Document.getRootElement()** - Get the root element of the XML.
* **Element.getName()** - Get the name of the XML node.
* **Element.getChildren()** - Get all the direct child nodes of an element.
* **Node.getChildren(Name)** - Get all the direct child nodes with a given name.
* **Node.getChild(Name)** - Get first child node with given name.

# Java JDOM Parser - Parse XML Document

## Steps to Using JDOM

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

* Import XML-related packages.
* Create a SAXBuilder
* Create a Document from a file or stream
* Extract the root element
* Examine attributes
* Examine sub-elements

**Import XML-related packages**

import java.io.\*;

import java.util.\*;

import org.jdom2.\*;

**Create a DocumentBuilder**

SAXBuilder saxBuilder = new SAXBuilder();

**Create a Document from a file or stream**

File inputFile = new File("input.txt");

SAXBuilder saxBuilder = new SAXBuilder();

Document document = saxBuilder.build(inputFile);

**Extract the root element**

Element classElement = document.getRootElement();

**Examine attributes**

//returns specific attribute

getAttribute("attributeName");

**Examine sub-elements**

//returns a list of subelements of specified name

getChildren("subelementName");

//returns a list of all child nodes

getChildren();

//returns first child node

getChild("subelementName");

## Demo Example

**Here is the input xml file 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>

**Demo Example:**

*DomParserDemo.java*

import java.io.File;

import java.io.IOException;

import java.util.List;

import org.jdom2.Attribute;

import org.jdom2.Document;

import org.jdom2.Element;

import org.jdom2.JDOMException;

import org.jdom2.input.SAXBuilder;

public class JDomParserDemo {

public static void main(String[] args) {

try {

File inputFile = new File("input.txt");

SAXBuilder saxBuilder = new SAXBuilder();

Document document = saxBuilder.build(inputFile);

System.out.println("Root element :"

+ document.getRootElement().getName());

Element classElement = document.getRootElement();

List<Element> studentList = classElement.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(JDOMException e){

e.printStackTrace();

}catch(IOException ioe){

ioe.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

# Java JDOM Parser - Query XML Document

## Demo Example

**Here is the input xml file we need to query:**

<?xml version="1.0"?>

<cars>

<supercars company="Ferrari">

<carname type="formula one">Ferarri 101</carname>

<carname type="sports car">Ferarri 201</carname>

<carname type="sports car">Ferarri 301</carname>

</supercars>

<supercars company="Lamborgini">

<carname>Lamborgini 001</carname>

<carname>Lamborgini 002</carname>

<carname>Lamborgini 003</carname>

</supercars>

<luxurycars company="Benteley">

<carname>Benteley 1</carname>

<carname>Benteley 2</carname>

<carname>Benteley 3</carname>

</luxurycars>

</cars>

**Demo Example:**

*QueryXmlFileDemo.java*

import java.io.File;

import java.io.IOException;

import java.util.List;

import org.jdom2.Attribute;

import org.jdom2.Document;

import org.jdom2.Element;

import org.jdom2.JDOMException;

import org.jdom2.input.SAXBuilder;

public class QueryXmlFileDemo {

public static void main(String[] args) {

try {

File inputFile = new File("input.txt");

SAXBuilder saxBuilder = new SAXBuilder();

Document document = saxBuilder.build(inputFile);

System.out.println("Root element :"

+ document.getRootElement().getName());

Element classElement = document.getRootElement();

List<Element> supercarList = classElement.getChildren("supercars");

System.out.println("----------------------------");

for (int temp = 0; temp < supercarList.size(); temp++) {

Element supercarElement = supercarList.get(temp);

System.out.println("\nCurrent Element :"

+ supercarElement.getName());

Attribute attribute = supercarElement.getAttribute("company");

System.out.println("company : "

+ attribute.getValue() );

List<Element> carNameList = supercarElement.getChildren("carname");

for (int count = 0;

count < carNameList.size(); count++) {

Element carElement = carNameList.get(count);

System.out.print("car name : ");

System.out.println(carElement.getText());

System.out.print("car type : ");

Attribute typeAttribute = carElement.getAttribute("type");

if(typeAttribute !=null)

System.out.println(typeAttribute.getValue());

else{

System.out.println("");

}

}

}

}catch(JDOMException e){

e.printStackTrace();

}catch(IOException ioe){

ioe.printStackTrace();

}

}

}

This would produce the following result:

Root element :cars

----------------------------

Current Element :supercars

company : Ferrari

car name : Ferarri 101

car type : formula one

car name : Ferarri 201

car type : sports car

car name : Ferarri 301

car type : sports car

Current Element :supercars

company : Lamborgini

car name : Lamborgini 001

car type :

car name : Lamborgini 002

car type :

car name : Lamborgini 003

car type :

# Java JDOM Parser - Create XML Document

## Demo Example

**Here is the XML we need to create:**

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

<cars>

<supercars company="Ferrari">

<carname type="formula one">Ferrari 101</carname>

<carname type="sports">Ferrari 202</carname>

</supercars>

</cars>

**Demo Example:**

*CreateXmlFileDemo.java*

import java.io.IOException;

import org.jdom2.Attribute;

import org.jdom2.Document;

import org.jdom2.Element;

import org.jdom2.output.Format;

import org.jdom2.output.XMLOutputter;

public class CreateXmlFileDemo {

public static void main(String[] args) {

try{

//root element

Element carsElement = new Element("cars");

Document doc = new Document(carsElement);

//supercars element

Element supercarElement = new Element("supercars");

supercarElement.setAttribute(new Attribute("company","Ferrari"));

//supercars element

Element carElement1 = new Element("carname");

carElement1.setAttribute(new Attribute("type","formula one"));

carElement1.setText("Ferrari 101");

Element carElement2 = new Element("carname");

carElement2.setAttribute(new Attribute("type","sports"));

carElement2.setText("Ferrari 202");

supercarElement.addContent(carElement1);

supercarElement.addContent(carElement2);

doc.getRootElement().addContent(supercarElement);

XMLOutputter xmlOutput = new XMLOutputter();

// display ml

xmlOutput.setFormat(Format.getPrettyFormat());

xmlOutput.output(doc, System.out);

}catch(IOException e){

e.printStackTrace();

}

}

}

This would produce the following result:

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

<cars>

<supercars company="Ferrari">

<carname type="formula one">Ferrari 101</carname>

<carname type="sports">Ferrari 202</carname>

</supercars>

</cars>

# Java JDOM Parser - Modify XML Document

## Demo Example

**Here is the input text file we need to modify:**

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<cars>

<supercars company="Ferrari">

<carname type="formula one">Ferrari 101</carname>

<carname type="sports">Ferrari 202</carname>

</supercars>

<luxurycars company="Benteley">

<carname>Benteley 1</carname>

<carname>Benteley 2</carname>

<carname>Benteley 3</carname>

</luxurycars>

</cars>

**Demo Example:**

*ModifyXmlFileDemo.java*

import java.io.File;

import java.io.IOException;

import java.util.List;

import org.jdom2.Attribute;

import org.jdom2.Document;

import org.jdom2.Element;

import org.jdom2.JDOMException;

import org.jdom2.input.SAXBuilder;

import org.jdom2.output.Format;

import org.jdom2.output.XMLOutputter;

public class ModifyXMLFileDemo {

public static void main(String[] args) {

try {

File inputFile = new File("input.txt");

SAXBuilder saxBuilder = new SAXBuilder();

Document document = saxBuilder.build(inputFile);

Element rootElement = document.getRootElement();

//get first supercar

Element supercarElement = rootElement.getChild("supercars");

// update supercar attribute

Attribute attribute = supercarElement.getAttribute("company");

attribute.setValue("Lamborigini");

// loop the supercar child node

List<Element> list = supercarElement.getChildren();

for (int temp = 0; temp < list.size(); temp++) {

Element carElement = list.get(temp);

if("Ferrari 101".equals(carElement.getText())){

carElement.setText("Lamborigini 001");

}

if("Ferrari 202".equals(carElement.getText())){

carElement.setText("Lamborigini 002");

}

}

//get all supercars element

List<Element> supercarslist = rootElement.getChildren();

for (int temp = 0; temp < supercarslist.size(); temp++) {

Element tempElement = supercarslist.get(temp);

if("luxurycars".equals(tempElement.getName())){

rootElement.removeContent(tempElement);

}

}

XMLOutputter xmlOutput = new XMLOutputter();

// display xml

xmlOutput.setFormat(Format.getPrettyFormat());

xmlOutput.output(document, System.out);

} catch (JDOMException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}

This would produce the following result:

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

<cars>

<supercars company="Lamborigini">

<carname type="formula one">Lamborigini 001</carname>

<carname type="sports">Lamborigini 002</carname>

</supercars>

</cars>