The Document Object Model 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 the DOM implement that 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 the document around (you might want to sort certain elements, for example)
* You need to use the information in the 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 changes.

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 the 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, returns the attribute with the requested name.

# Java DOM Parser - Parse XML Document

Steps to Using DOM

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

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

**Import XML-related packages**

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import java.io.\*;

**Create a DocumentBuilder**

DocumentBuilderFactory factory =

DocumentBuilderFactory.newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();

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

StringBuilder xmlStringBuilder = new StringBuilder();

xmlStringBuilder.append("<?xml version="1.0"?> <class> </class>");

ByteArrayInputStream input = new ByteArrayInputStream(

xmlStringBuilder.toString().getBytes("UTF-8"));

Document doc = builder.parse(input);

**Extract the root element**

Element root = document.getDocumentElement();

**Examine attributes**

//returns specific attribute

getAttribute("attributeName");

//returns a Map (table) of names/values

getAttributes();

**Examine sub-elements**

//returns a list of subelements of specified name

getElementsByTagName("subelementName");

//returns a list of all child nodes

getChildNodes();

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*

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

# Java DOM 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*

package com.tutorialspoint.xml;

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;

import java.io.File;

public class QueryXmlFileDemo {

public static void main(String argv[]) {

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.print("Root element: ");

System.out.println(doc.getDocumentElement().getNodeName());

NodeList nList = doc.getElementsByTagName("supercars");

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

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = nList.item(temp);

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

System.out.print(nNode.getNodeName());

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

System.out.print("company : ");

System.out.println(eElement.getAttribute("company"));

NodeList carNameList =

eElement.getElementsByTagName("carname");

for (int count = 0;

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

Node node1 = carNameList.item(count);

if (node1.getNodeType() ==

node1.ELEMENT\_NODE) {

Element car = (Element) node1;

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

System.out.println(car.getTextContent());

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

System.out.println(car.getAttribute("type"));

}

}

}

}

} catch (Exception e) {

e.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 DOM Parser - Create XML Document

## Demo Example

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

<?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></cars>

**Demo Example:**

*CreateXmlFileDemo.java*

package com.tutorialspoint.xml;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Attr;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import java.io.File;

public class CreateXmlFileDemo {

public static void main(String argv[]) {

try {

DocumentBuilderFactory dbFactory =

DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder =

dbFactory.newDocumentBuilder();

Document doc = dBuilder.newDocument();

// root element

Element rootElement = doc.createElement("cars");

doc.appendChild(rootElement);

// supercars element

Element supercar = doc.createElement("supercars");

rootElement.appendChild(supercar);

// setting attribute to element

Attr attr = doc.createAttribute("company");

attr.setValue("Ferrari");

supercar.setAttributeNode(attr);

// carname element

Element carname = doc.createElement("carname");

Attr attrType = doc.createAttribute("type");

attrType.setValue("formula one");

carname.setAttributeNode(attrType);

carname.appendChild(

doc.createTextNode("Ferrari 101"));

supercar.appendChild(carname);

Element carname1 = doc.createElement("carname");

Attr attrType1 = doc.createAttribute("type");

attrType1.setValue("sports");

carname1.setAttributeNode(attrType1);

carname1.appendChild(

doc.createTextNode("Ferrari 202"));

supercar.appendChild(carname1);

// write the content into xml file

TransformerFactory transformerFactory =

TransformerFactory.newInstance();

Transformer transformer =

transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result =

new StreamResult(new File("C:\\cars.xml"));

transformer.transform(source, result);

// Output to console for testing

StreamResult consoleResult =

new StreamResult(System.out);

transformer.transform(source, consoleResult);

} catch (Exception e) {

e.printStackTrace();

}

}

}

This would produce the following result:

<?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>

# Java DOM Parser - Modify XML Document

## Demo Example

**Here is the input xml 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*

package com.tutorialspoint.xml;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NamedNodeMap;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

public class ModifyXmlFileDemo {

public static void main(String argv[]) {

try {

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

DocumentBuilderFactory docFactory =

DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder =

docFactory.newDocumentBuilder();

Document doc = docBuilder.parse(inputFile);

Node cars = doc.getFirstChild();

Node supercar = doc.getElementsByTagName("supercars").item(0);

// update supercar attribute

NamedNodeMap attr = supercar.getAttributes();

Node nodeAttr = attr.getNamedItem("company");

nodeAttr.setTextContent("Lamborigini");

// loop the supercar child node

NodeList list = supercar.getChildNodes();

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

Node node = list.item(temp);

if (node.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) node;

if ("carname".equals(eElement.getNodeName())){

if("Ferrari 101".equals(eElement.getTextContent())){

eElement.setTextContent("Lamborigini 001");

}

if("Ferrari 202".equals(eElement.getTextContent()))

eElement.setTextContent("Lamborigini 002");

}

}

}

NodeList childNodes = cars.getChildNodes();

for(int count = 0; count < childNodes.getLength(); count++){

Node node = childNodes.item(count);

if("luxurycars".equals(node.getNodeName()))

cars.removeChild(node);

}

// write the content on console

TransformerFactory transformerFactory =

TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

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

StreamResult consoleResult = new StreamResult(System.out);

transformer.transform(source, consoleResult);

} catch (Exception e) {

e.printStackTrace();

}

}

}

This would produce the following result:

-----------Modified File-----------

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

<cars>

<supercars company="Lamborigini">

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

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

</supercars></cars>