Java DOM Parser - Parse XML Document

Java DOM parser is a Java API to parse any XML document. Using the methods provided, we can retrieve root element, sub elements and their attributes using Java DOM parser.

In this tutorial we have used the **getTagName()** method to retrieve the tag name of elements, getFirstChild() to retrieve the first child of an element and getTextContent() to get the text content of elements.

Parse XML Using Java DOM parser

Having discussed various XML parsers available in Java, now let us see how we can use DOM parser to parse an XML file. We use **parse()** method to parse an XML file. Before jumping into the example directly, let us see the steps to parse XML document using Java DOM parser –

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

Step 1: Creating a DocumentBuilder Object

DocumentBuilderFactory is a factory API to obtain parser to parse XML documents by creating DOM trees. It has 'newDocumentBuilder()' method that creates an instance of the class 'DocumentBuilder'. This DocumentBuilder class is used to get input in the form of streams, files, URLs and SAX InputSources.

```
DocumentBuilderFactory factory =DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder = factory.newDocumentBuilder();
```

Step 2: Reading the XML

Input can be of file type or stream type. To input an XML file, Create a file object and pass the file path as argument.

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

To get input in the form of stream, we have used StringBuilder class and appended the input string and later converted it into bytes. The obtained ByteArrayInputStream is given as input to the document.

```
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

DocumentBuilder created in above steps is used to parse the input XML file. It contains a method named parse() which accepts a file or input stream as a parameter and returns a DOM Document object. If the given file or input stream is NULL, this method throws an IllegalArgumentException.

```
Document xmldoc = docBuilder.parse(input);
```

Step4: Retrieving the Elements

The Node and Element interfaces of the **org.w3c.dom.** package provides various methods to retrieve desired information about elements from the XML documents. This information includes element's name, text content, attributes and their values. We have many DOM interfaces and methods to get this information.

Retrieving Root Element Name

XML document constitutes of many elements. In Java an XML/HTML document is represented by the interface named **Element**. This interface provides various methods to retrieve, add and modify the contents of an XML/HTML document.

We can retrieve the name of the root element using the method named **getTagName()** of the Element interface. It returns the name of the root element in the form of a string.

Since Element is an interface, to create its object we need to use the **getDocumentElement()** method. This method retrieves and returns the root element in the form of an object.

Example

In the following example we have passed a simple XML document with just one root element 'college' using StringBuilder class. Then, we are retrieving it and printing on the console.

</> Open Compiler

```
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import java.io.ByteArrayInputStream;
import javax.xml.parsers.DocumentBuilder;

public class RetrieveRootElementName {
   public static void main(String[] args) {
      try {
         //Creating a DocumentBuilder Object
         DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
         DocumentBuilder docBuilder = factory.newDocumentBuilder();

         //Reading the XML
         StringBuilder xmlBuilder = new StringBuilder();
         xmlBuilder.append("<college></college>");

         //Parsing the XML Document
         ByteArrayInputStream input = new
ByteArrayInputStream(xmlBuilder.toString().getBytes("UTF-8"));
         Document xmldoc = docBuilder.parse(input);

         //Retrieving the Root Element Name
         Element element = xmldoc.getDocumentElement();
         System.out.println("Root element name is "+element.getTagName());

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

Output

The element name, 'college' is displayed on the output screen as shown below –

```
Root element name is college
```

Parsing Single Sub Element in XML

We can parse a simple XML document with single element inside the root element. So far, we have seen how to retrieve the root element. Now, let us see how to get the sub element inside the root element.

Since, we have only one sub element, we are using **getFirstChild()** method to retrieve it. This method is used with the root element to get its first child. It returns the child node in the form of a Node object.

After retrieving the child node, **getNodeName()** method is used to get the name of the node. It returns the node name in the form of a string.

To get the text content, we use **getTextContent()** method. It returns the text content in the form of a String.

Example

Let us see the following example where we have one root element and a sub element. Here, 'college' is the root element with 'department' as sub element. The 'department' element has text content, "Computer Science". We are retrieving the name and text content of the sub element.

</> Open Compiler

```
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;

import java.io.ByteArrayInputStream;
import javax.xml.parsers.DocumentBuilder;

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

      try {

         //Creating a DocumentBuilder Object
         DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
```

```
         DocumentBuilder docBuilder = factory.newDocumentBuilder();

         //Reading the XML
         StringBuilder xmlBuilder = new StringBuilder();
         xmlBuilder.append("<college><department>Computer Science</department>
</college>");

         //Parsing the XML Document
         ByteArrayInputStream input = new
ByteArrayInputStream(xmlBuilder.toString().getBytes("UTF-8"));
         Document xmldoc = docBuilder.parse(input);

         //Retrieving the Root Element
         Element element = xmldoc.getDocumentElement();

         //Retrieving the Child Node
         Node childNode = element.getFirstChild();
         String childNodeName = childNode.getNodeName();
         System.out.println("Sub Element name : " + childNodeName);
         //Retrieving Text Content of the Child Node "+ childNodeName);

         System.out.println("Text content of Sub Element :
"+childNode.getTextContent());

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

The output window displays Sub element name and text content.

```
Sub Element name : department
Text content of Sub Element : Computer Science
```

Parsing Multiple Elements in XML

To parse an XML document with multiple elements we need to use loops. The **getChildNodes()** method retrieves all the child nodes of an element and returns it as a NodeList. We need to loop through all the elements of the obtained NodeList and retrieve the desired information about each element as we did in the previous sections.

Example

Now, let us add two more departments to the XML file (**multipleElements.xml**). Let us try to retrieve all the department names and staff count.

```
<college>
   <department>
      <name>Computer Science</name>
      <staffCount>20</staffCount>
   </department>
      <department>
      <name>Electrical and Electronics</name>
      <staffCount>23</staffCount>
   </department>
      <department>
      <name>Mechanical</name>
      <staffCount>15</staffCount>
   </department>
</college>
```

In the following program, we retrieve the list of department elements into a NodeList and iterate all the departments to get the department name and staff count.

```
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 MultipleElementsXmlParsing {
   public static void main(String[] args) {
      try {

         //Input the XML file
         File inputXmlFile = new File("src/multipleElements.xml");

         //creating DocumentBuilder
         DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
```

```
         DocumentBuilder docBuilder = dbFactory.newDocumentBuilder();
         Document xmldoc = docBuilder.parse(inputXmlFile);

         //Retrieving the Root Element
         Element element = xmldoc.getDocumentElement();
         System.out.println("Root element name is "+element.getTagName());

         //Getting the child elements List
         NodeList nList = element.getChildNodes();

         //Iterating through all the child elements of the root
         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("Name of the department : " +
eElement.getElementsByTagName("name").item(0).getTextContent());
               System.out.println("Staff Count of the department : " +
eElement.getElementsByTagName("staffCount").item(0).getTextContent());
            }
         }
      } catch (Exception e) {
            e.printStackTrace();
      }
   }
}
```

All the three departments with name and staff count are displayed.

```
Root element :college

Current Element :department
Name of the department : Computer Science
Staff Count of the department : 20

Current Element :department
Name of the department : Electrical and Electronics
Staff Count of the department : 23

Current Element :department
```

```
Name of the department : Mechanical
Staff Count of the department : 15
```

Parsing Attributes in XML

XML elements can have attributes and these can be retrieved using the **getAttribute()** method. This method takes attribute name as a parameter and returns its corresponding attribute value as a String. It returns an empty string if there is no attribute value or default value for the attribute name specified.

Example

Now, let us add an attribute, 'deptcode' to all the department elements in the '**attributesParsing.xml**' file.

```
<?xml version = "1.0"?>
<college>
   <department deptcode = "DEP_CS23">
      <name>Computer Science</name>
      <staffCount>20</staffCount>
   </department>

   <department deptcode = "DEP_EC34">
      <name>Electrical and Electronics</name>
      <staffCount>23</staffCount>
   </department>

   <department deptcode = "DEP_MC89">
      <name>Mechanical</name>
      <staffCount>15</staffCount>
   </department>
</college>
```

In the following program, we are retrieving deptcode along with name and staff count for each department.

```
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 AttributesXmlParsing {
   public static void main(String[] args) {
      try {
         //Input the XML file
         File inputXmlFile = new File("attributesParsing.xml");

         //creating DocumentBuilder
         DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
         DocumentBuilder docBuilder = dbFactory.newDocumentBuilder();
         Document xmldoc = docBuilder.parse(inputXmlFile);

         //Getting the root element
         System.out.println("Root element :" +
xmldoc.getDocumentElement().getNodeName());
         NodeList nList = xmldoc.getElementsByTagName("department");

         //Iterating through all the child elements of the root
         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("Department Code : " +
eElement.getAttribute("deptcode"));
               System.out.println("Name of the department : " +
eElement.getElementsByTagName("name").item(0).getTextContent());
               System.out.println("Staff Count of the department : " +
eElement.getElementsByTagName("staffCount").item(0).getTextContent());
            }
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

The three departments are displayed with their corresponding department code, name and staff count.

```
Root element :college

Current Element :department
Department Code : DEP_CS23
Name of the department : Computer Science
Staff Count of the department : 20

Current Element :department
Department Code : DEP_EC34
Name of the department : Electrical and Electronics
Staff Count of the department : 23

Current Element :department
Department Code : DEP_MC89
Name of the department : Mechanical
Staff Count of the department : 15
```