Java JDOM Parser - Modify XML Document

Java JDOM parser is an API in java that has classes and interfaces to modify XML documents. This API represents the XML document in the form of a tree structure and each element can be retrieved easily. Hence, modification becomes an easy task. We can use setText() to modify content and addContent() to add new elements. In this chapter, we are going to see how to modify an existing XML document with two examples.

Modify XML using JDOM Parser

We can modify an XML document in java JDOM parser through following steps –

- **Step 1:** Creating a SAXBuilder Object
- **Step 2:** Reading the XML
- **Step 3:** Parsing the XML Document
- **Step 4:** Updating the content of XML document
- **Step 5:** Writing the content into XML file
- **Step 6:** Output to console for testing

Refer second chapter of this section for first three steps.

Step 4: Updating the content of XML document

After following the first three steps, we have successfully read the XML document we need to update. Now, we have the XML file in the form of JDOM document. To get any information from the document, firstly, we should always get the root element. After retrieving the root element, we can get the information of all the elements inside the root.

Refer this chapter of this section for steps 5 and 6.

Updating Text Content

To update text content of any element, we can use the **setText()** method of Element class. This method takes the text content as an argument in the form of a String. If text content already exists, it updates the old one with the new text content.

Example

Here, we have **college.xml** file that has three department elements. We are going to modify the staff count for department with id, 102 from 23 to 14.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><college>
   <department id="101">
      <name>Computer Science</name>
      <staffCount>20</staffCount>
   </department>
   <department id="102">
      <name>Electrical and Electronics</name>
      <staffCount>23</staffCount>
   </department>
   <department id="103">
      <name>Mechanical</name>
      <staffCount>15</staffCount>
   </department>
</college>
```

The following **ModifyTextContent.java** program reads the above college.xml file using SAXBuilder object. After getting the list of department elements, we are using getAttributeValue() method to find the department with id, 102. Later, we are updating the text content using setText() method.

```
import java.io.File;
import java.util.List;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import org.jdom2.Document;
import org.jdom2.Element;
import org.jdom2.input.SAXBuilder;
import org.jdom2.output.Format;
import org.jdom2.output.XMLOutputter;
import org.jdom2.transform.JDOMSource;

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

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

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

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

         //Retrieving the Root Element
         Element RootElement = doc.getRootElement();
         List<Element> deptList = RootElement.getChildren("department");

         //Finding "department" with id=102 in the list
         for(int index=0; index<deptList.size();index++) {
            Element department = deptList.get(index);
            if(department.getAttributeValue("id").equals("102")) {
               Element staffCount = department.getChild("staffCount");
               //Updating the staff count
               staffCount.setText("14");
               break;
            }
         }
         //writing the modified content into XML file
         TransformerFactory transformerFactory =
TransformerFactory.newInstance();
         Transformer transformer = transformerFactory.newTransformer();
         JDOMSource source = new JDOMSource(doc);
         StreamResult result = new StreamResult(new File("college.xml"));
         transformer.transform(source, result);

         //Output to console for testing
         XMLOutputter xmlOutput = new XMLOutputter();
         xmlOutput.setFormat(Format.getPrettyFormat());
         xmlOutput.output(doc, System.out);
      } catch(Exception e) {
            e.printStackTrace();
      }
   }
}
```

Output

Following is the updated XML document after updating staffCount.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><college>
   <department id="101">
```

```
      <name>Computer Science</name>
      <staffCount>20</staffCount>
   </department>
   <department id="102">
      <name>Electrical and Electronics</name>
      <staffCount>14</staffCount>
   </department>
   <department id="103">
      <name>Mechanical</name>
      <staffCount>15</staffCount>
   </department>
</college>
```


Adding New Elements

The **addContent()** method of Element class takes child element and append it one level deep to the current element. It always adds the new Element at the end. If we pass two arguments, index and child element, then the child element gets inserted at the specified index.

Example

Now, we are going to add one more department named "Civil" to our college element in the following **AddNewElements.java** program. We have created the child elements of the department and added them to the department element. Later we added the department to the root element.

```
import java.io.File;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import org.jdom2.Document;
import org.jdom2.Element;
import org.jdom2.input.SAXBuilder;
import org.jdom2.output.Format;
import org.jdom2.output.XMLOutputter;
import org.jdom2.transform.JDOMSource;

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

```
      try {

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

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

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

         //Retrieving the Root Element
         Element RootElement = doc.getRootElement();

         //Creating new "department" Element
         Element department=new Element("department");
         department.setAttribute("id","104");

         //Creating "name" Element for department
         Element name=new Element("name");
         name.setText("Civil");

         //Creating "staffCount" Element for department
         Element staffCount=new Element("staffCount");
         staffCount.setText("18");

         //Appending Elements
         department.addContent(name);
         department.addContent(staffCount);
         RootElement.addContent(department);

         //writing the modified content into XML file
         TransformerFactory transformerFactory =
TransformerFactory.newInstance();
         Transformer transformer = transformerFactory.newTransformer();
         JDOMSource source = new JDOMSource(doc);
         StreamResult result = new StreamResult(new File("college.xml"));
         transformer.transform(source, result);

         //Output to console for testing
         XMLOutputter xmlOutput = new XMLOutputter();
         xmlOutput.setFormat(Format.getPrettyFormat());
```

```
            xmlOutput.output(doc, System.out);
         } catch(Exception e) {
               e.printStackTrace();
         }
      }
}
```

Output

The updated file after adding "Civil" department is as follows :

```
<?xml version="1.0" encoding="UTF-8"?>
<college>
  <department id="101">
    <name>Computer Science</name>
    <staffCount>20</staffCount>
  </department>
  <department id="102">
    <name>Electrical and Electronics</name>
    <staffCount>23</staffCount>
  </department>
  <department id="103">
    <name>Mechanical</name>
    <staffCount>15</staffCount>
  </department>
  <department id="104">
    <name>Civil</name>
    <staffCount>18</staffCount>
  </department>
</college>
```