Java JDOM Parser - Create XML Document

Java JDOM Parser is a Java API that has classes and methods to create XML documents from scratch. We can create a new JDOM document object and add elements, attributes using the methods available in Document and Element interfaces. In this chapter, we are going to use setRootElement(), addContent(), setText() and setAttribute() methods to create XML files in detail.

Create XML using Java JDOM parser

We can create an XML document in Java using JDOM parser through following steps –

- **Step 1:** Creating JDOM Document object
- **Step 2:** Creating and appending Root Element
- **Step 3:** Creating elements and attributes
- **Step 4:** Appending Elements to Root
- **Step 5:** Writing the content into XML file
- **Step 6:** Testing the output using console

Step 1: Creating JDOM Document object

The **org.jdom2** package has Document class. It represents the XML document. This class has methods to access the root element and also the document level information. We create a new document as follows –

```
Document doc = new Document();
```

Step 2: Creating and appending Root Element

An XML document must contain a root element. We create root element using Element class of org.jdom2 package. If we provide a String to the constructor, it creates element with that supplied local name.

The setRootElement() method in the Document class sets the root of the document. This method takes Element as an argument and sets it as the root. If the root element is already present, the old root Element gets replaced with this supplied Element.

```
Element RootElement = new Element("root");
```

```
doc.setRootElement(RootElement);
```

Step 3: Creating elements and attributes

We can create Elements the same way we created root element in the previous step. To set an attribute to the Element, we use setAttribute() method as follows −

```
Element newElement = new Element("FirstElement");
newElement.setAttribute("attr_name","attr_value");
```

Step 4: Appending Elements to Root

The Elements created in the previous step are now attached to the root element using addContent() method. This method takes single element or collection of elements in the form of content list and adds them to the element accordingly.

```
RootElement.addContent(newElement);
```

Step 5: Writing the content into XML file

The TransformerFactory class is used to create a new instance of Transformer object. Using the transform() function in Transformer class, source is transformed to the destination result. We are creating JDOMSource object by passing document as a parameter. This JDOMSource object is transformed into StreamResult as follows −

```
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
JDOMSource source = new JDOMSource(doc);
StreamResult result = new StreamResult(new File("newFile.xml"));
transformer.transform(source, result);
```

Step 6: Testing the output using console

This is an optional step used for testing purpose. To print the output on the console, an XMLOutputter object is created and the Document object is passed as follows −

```
XMLOutputter xmlOutput = new XMLOutputter();
xmlOutput.setFormat(Format.getPrettyFormat());
xmlOutput.output(doc, System.out);
```

Creating Simple XML File

Using, the above mentioned steps, let us create a simple XML file that has root element alone. A new document object is created and the Root element is attached using **setRootElement()** method.

The **setText()** method of Element object takes the text content in the form of a String and attaches it to the Element.

Example

Here is the XML file we need to create. It has a root element named "cars" and the text content, "Ferrari".

```
<cars>Ferrari</cars>
```

The following CreateXMLFile.java creates the XML file and stores it in D drive with the file name as "cars.xml".

```
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.output.Format;
import org.jdom2.output.XMLOutputter;
import org.jdom2.transform.JDOMSource;

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

         //Creating JDOM Document object
         Document doc = new Document();

         //Creating and appending Root Element
         Element carsElement = new Element("cars");
         carsElement.setText("Ferrari");
         doc.setRootElement(carsElement);

         //writing the content into XML file
```

```
         TransformerFactory transformerFactory =
TransformerFactory.newInstance();
         Transformer transformer = transformerFactory.newTransformer();
         JDOMSource source = new JDOMSource(doc);
         StreamResult result = new StreamResult(new File("D:\\cars.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();
      }
   }
}
```

For testing purpose, we have printed the content of XML document on the console.

```
<?xml version="1.0" encoding="UTF-8"?>
<cars>Ferrari</cars>
```

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

Creating Attributes

Let us now add child elements to the root Element. Also, let us add attributes to our elements. In this example, we see how to create elements along with their attributes and attach them to the root. The **setAttribute()** method on each element sets the attribute and setText() method is used to set the text content of each element.

Example

Here is the cars.xml file that we need to create –

```
<cars>
<supercars company="Ferrari">
<carname type="formula one">Ferrari 101</carname>
<carname type="sports">Ferrari 202</carname>
</supercars>
</cars>
```

The following **CreateAttributes.java** program creates the cars.xml file in D drive.

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

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

         //Creating JDOM Document object
         Document doc = new Document();

         //Creating and appending Root Element
         Element carsElement = new Element("cars");
         doc.setRootElement(carsElement);

         //Creating elements and attributes
         Element supercarElement = new Element("supercars");
         supercarElement.setAttribute("company","Ferrari");

         Element carElement1 = new Element("carname");
         carElement1.setAttribute("type","formula one");
         carElement1.setText("Ferrari 101");

         Element carElement2 = new Element("carname");
         carElement2.setAttribute("type","sports");
         carElement2.setText("Ferrari 202");

         supercarElement.addContent(carElement1);
         supercarElement.addContent(carElement2);

         //Appending Elements to Root
         carsElement.addContent(supercarElement);
```

```
         //writing the content into XML file
         TransformerFactory transformerFactory =
TransformerFactory.newInstance();
         Transformer transformer = transformerFactory.newTransformer();
         JDOMSource source = new JDOMSource(doc);
         StreamResult result = new StreamResult(new File("D:\\Jdomcars.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();
      }
   }
}
```

The output window displays the file content as follows –

```
<?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>
```