Java DOM Parser - Create XML Document

Java DOM parser API has methods, interfaces and classes to create XML documents. Using this API, we can create XML documents from the scratch through our Java applications. The createElement() method creates new elements and the appendChild() method appends the created elements to already existing elements.

Create XML Using Java DOM Parser

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

- **Step 1:** Creating a DocumentBuilder Object
- **Step 2:** Create a new Document
- **Step 3:** Creating the root element
- **Step 4:** Appending elements to the root element
- **Step 5:** Writing the content into XML file
- **Step 6:** Testing the output using console

Refer this page for step 1.

Step2: Create a new Document

Using DocumentBuilder object created in the above step, create a new document with newDocument() function.

```
Document doc = dBuilder.newDocument();
```

Step3: Creating the root element

Every XML document should possess a single root element , which is also called a parent element. We can create the root element using createElement() method and append it to the document created in the previous step.

```
Element rootElement = doc.createElement("root");
doc.appendChild(rootElement);
```

Step 4: Appending elements to the root element

Inside the root element, we can create any number of child elements and append them the same way we appended root element to the document. To write text content inside an element, we can use createTextNode() method. We can also create attributes for elements using createAttribute() method.

```
Element child = doc.createElement("child");
rootElement.appendChild(child);
child.appendChild(doc.createTextNode("text_content_here"));
Attr attr = doc.createAttribute("child");
```

Step 5: Writing the content into XML file

After building the elements inside the document with their corresponding attributes, we need to write this content into an XML file by creating Tranformer object, which in-turn transforms our source document into StreamResult and stores it in the specified file path with the given name.

```
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
DOMSource source = new DOMSource(doc);
StreamResult result = new StreamResult(new File("filepath:\\new.xml"));
transformer.transform(source, result);
```

Step 6: Testing the output using console

We can test our XML file by printing it on the console. This is an optional step.

```
StreamResult consoleResult = new StreamResult(System.out);
transformer.transform(source, consoleResult);
```

Creating Basic XML File

To create a document element, we use the method **createElement("element_name")**. This method takes element name as an argument and returns a new Element object.

After getting the new Element object, we can use **appendChild(Element)** method to append this Element object to the document. This method is also used to append an element to another element (creating child elements).

cars.xml

We want to create the following xml file with the name "cars.xml" in D drive.

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

CreateXMLDemo.java

In the following code, we have created one root element with name "cars" and appended one child element with name "supercars". We have also added text content to supercars element.

```
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.Document;
import org.w3c.dom.Element;
import java.io.File;

public class CreateXMLDemo {

   public static void main(String argv[]) {

      try {

         //Creating a DocumentBuilder Object
         DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

         //Create a new Document
         Document doc = dBuilder.newDocument();

         //Creating the root element
         Element rootElement = doc.createElement("cars");
         doc.appendChild(rootElement);

         //Appending elements to the root element
```

```
         Element supercar = doc.createElement("supercars");
         rootElement.appendChild(supercar);
         supercar.appendChild(doc.createTextNode("Ferrari"));

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

Output

For testing, we have printed the document output on the console.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><cars><supercars>Ferrari</s
```

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

Creating XML File with Attributes

Attributes can be created to an Element using the method, **createAttribute("Attribute_name")**. This method of Document interface takes name of the attribute as an argument and returns the **Attr** object. The **setValue("value")** method is used to set the value of the attribute.

Now, to set this attribute node to the Element, we use **setAttributeNode(Attr)** method with the Element object to which we need to set this attribute.

newcars.xml

We need to create the following "newcars.xml" in D drive.

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

CreateXMLAttributeDemo.java

We have created two carname elements for supercars element. Also, using setAttributeNode() method, we have added two attributes for each of the carname elements.

```
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 CreateXMLAttributeDemo {

   public static void main(String argv[]) {

      try {

         //Creating a DocumentBuilder Object
         DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

         //Creating a new Document
         Document doc = dBuilder.newDocument();

         //Creating the root element
         Element rootElement = doc.createElement("cars");
         doc.appendChild(rootElement);
```

```
         //Appending sub element to the root element
         Element supercar = doc.createElement("supercars");
         rootElement.appendChild(supercar);

         //Setting attribute to the sub element
         Attr attr = doc.createAttribute("company");
         attr.setValue("Ferrari");
         supercar.setAttributeNode(attr);

         //Adding First child element to sub 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);

         //Adding second child element to sub element
         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);

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

Output

For testing, we have printed the document output on the console.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><cars><supercars company="
```