Java XPath Parser - Parse XML Document

Java XPath parser is an API in Java to parse XML documents using XPath expressions and functions. It helps us to traverse through the entire XML document and obtain elements as nodes inside a NodeList. The package 'javax.xml.xpath' provides the API for the evaluation of XPath expressions. In this chapter, we will see how to traverse through all the nodes in an XML document.

Parse XML Using Java XPath Parser

Following are the steps used while parsing a document using Java XPath Parser –

- **Step 1:** Creating a DocumentBuilder
- **Step 2:** Reading the XML
- **Step 3:** Creating Document from file or Stream
- **Step 4:** Building XPath
- **Step 5:** Preparing and Evaluating XPath expression
- **Step 6:** Iterating over NodeList
- **Step 7:** Retrieving Elements

Step 1: Create a DocumentBuilder

The DocumentBuilderFactory is a factory API that is used to create DocumentBuilder objects. The newDocumentBuilder() method of DocumentBuilderFactory returns a DocumentBuilder object as follows –

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

Step 2: Reading the XML

The FileReader class is used to read streams of characters from the input file. The following statement throws "FileNotFoundException" if the file can't be found or if the file can't be read for some reason.

```
FileReader fileReader = new FileReader("src/input.txt");
```

Instead of reading XML content from the file, we can also get the content in the form of a string and convert it into bytes as follows –

```
StringBuilder xmlBuilder = new StringBuilder();
xmlBuilder.append("<class>xyz</class>");
ByteArrayInputStream input = new
ByteArrayInputStream(xmlBuilder.toString().getBytes("UTF-8"));
```

Step 3: Create a Document from a file or stream

The DocumentBuilder object created in the first step is used to create a document from the input file or input stream. The parse() method takes file or stream as an argument and returns a Document object as follows –

```
Document doc = builder.parse(input);
```

Step 4: Building XPath

To parse XML document using XPath, we need to build a newXPath using newXPath() method of XPathFactory. This method returns a new XPath as follows –

```
XPath xPath =  XPathFactory.newInstance().newXPath();
```

Step 5: Preparing and Evaluating XPath expression

As we have discussed in the previous chapter, XPath has expressions that help us retrieve information from the XML documents, here we need to create one such expression based on the requirement and evaluate it. The evaluate() method returns the result of the expression as a NodeList as follows –

```
String expression = "/class/student";
NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(
   doc, XPathConstants.NODESET);
```

Step 6: Iterating over NodeList

The NodeList we get in step 5 is now iterated to examine each node and retrieve information accordingly. Here, we have used a for loop to iterate over the NodeList, you can use any loop of your choice.

```
for (int i = 0; i < nodeList.getLength(); i++) {
   Node nNode = nodeList.item(i);
   ...
}
```

Step 7: Retrieving Elements

After following the above six steps, we obtain the elements in the form of nodes. By using the methods of interfaces available in DOM, we can retrieve the necessary elements and attributes.

Retrieve Root Element

To retrieve root element from the XML document, we have the XPath expression '**/**'. Using this expression and by evaluating this, we get the NodeList with just a single Node.

The **getNodeName()** method of Node interface retrieves the name of the node as a String object and the **getTextContent()** method returns the text content of the node as a String.

Example

In the following example, we have taken XML content as a StringBuilder object and parsed it using parse() function. We have only a single element, which is obviously the root and we have text content as 'xyz class'. Using the methods discussed above we retrieve the information of root element.

</> Open Compiler

```
import java.io.ByteArrayInputStream;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathFactory;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;

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

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

         //Reading the XML
         StringBuilder xmlBuilder = new StringBuilder();
         xmlBuilder.append("<class>xyz class</class>");
         ByteArrayInputStream input = new
ByteArrayInputStream(xmlBuilder.toString().getBytes("UTF-8"));

         //Creating Document from file or Stream
         Document doc = dBuilder.parse(input);

         //Building XPath
         XPath xPath =  XPathFactory.newInstance().newXPath();

         //Preparing and Evaluating XPath expression
         String expression = "/";
         NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(
            doc, XPathConstants.NODE);

         //Iterating over NodeList
         for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            //Retrieving Root Element
            System.out.println("Root Element Name: " + node.getNodeName());
            System.out.println("Text Content: " + node.getTextContent());
         }
      } catch(Exception e) {
          e.printStackTrace();
      }
   }
}
```

Output

The root element name and text content are displayed on the console.

```
Root Element Name: class
Text Content: xyz class
```


Retrieving Attributes

The NodeList we get after evaluating the XPath expression has nodes with different node types. We can convert those nodes into Elements if the node type is equal to 'ELEMENT_NODE'. The **getAttribute("attribute_name")** method of Element interface is used to retrieve the value of attribute in the form of a String.

Example

The following **studentData.xml** file has information of three students. We are going to retrieve this information using XPath parser library in Java. The class element is the root element with three student child elements. Let us see how to use XPath library in java to retrieve the infromation of three students.

```
<?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>singh</lastname>
      <nickname>jazz</nickname>
      <marks>90</marks>
```

```
   </student>
</class>
```

In the following **RetrieveAttributes.java** program, we have parsed the student.xml file and built a document. The expression **'/class/student'** is used to get all the 'student' nodes inside the 'class' node into a NodeList. The NodeList is then iterated and got the information of each student.

```
import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;

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

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

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

         //Creating Document from file or Stream
         Document doc = dBuilder.parse(inputFile);

         //Building XPath
         XPath xPath =  XPathFactory.newInstance().newXPath();

         //Preparing and Evaluating XPath expression
         String expression = "/class/student";
         NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(
            doc, XPathConstants.NODESET);
```

```
         //Iterating over NodeList
         for (int i = 0; i < nodeList.getLength(); i++) {
            Node nNode = nodeList.item(i);
            System.out.println("\nCurrent Element :" + nNode.getNodeName());
            //Retrieving Elements
            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();
      }
   }
}
```

Output

All the information of three students is displayed on the console.

```
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 : singh
Nick Name : jazz
Marks : 90
```