Java XPath Parser - Query XML Document

Java XPath is a Java API that helps us query XML documents using XPath expressions. Using XPath expressions, random access of elements is possible and hence we can select specific nodes from an XML document. This makes querying the documents more flexible.

In this chapter, we have used XPath expressions with predicates such as '/cars/carname/@company', '/class/student[@rollno = '493']' to query an XML document with various examples in detail.

Query XML Using Java XPath Parser

Following are the steps used while querying 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:** Querying Elements

Refer Parse XML Document chapter of this section for the first six steps.

Step 7: Querying Elements

Querying the XML document using XPath follows the same steps as parsing the XML document, but the only difference is seen in the way we prepare our XPath expressions. While querying, we query for a particular attribute or element.

Querying by Element Names

We can query the XML document based on their element names by specifying the element name inside the XPath expression. The XPath expression, **'root_element/element_name'** retrieves all the nodes with the specified name inside the root element.

Example

The following **cars.xml** file has information about seven cars with the element name as 'carname' inside the root element 'cars'. We are going to query this XML file to check if there is 'Bentley 2'car.

```
<?xml version = "1.0"?>
<cars>
      <carname company="Ferarri" >Ferarri 101</carname>
      <carname company="Lamborgini">Lamborgini 001</carname>
      <carname company="Lamborgini">Lamborgini 002</carname>
      <carname company="Lamborgini">Lamborgini 003</carname>
      <carname company="Bentley">Bentley 1</carname>
      <carname company="Bentley">Bentley 2</carname>
      <carname company="Bentley">Bentley 3</carname>
</cars>
```

The following **QueryByElements.java** program reads the above cars.xml file and builds a document. Using the XPath expression **'/cars/carname'**, we get the carname elements as nodes inside a nodeList. We iterate through the NodeList to find out Bentley 2 car.

```
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;

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

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

         //Reading the XML
         File inputFile = new File("cars.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 = "/cars/carname";
            NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(
               doc, XPathConstants.NODESET);

            //Iterating over NodeList
            boolean found = false;
            for (int i = 0; i < nodeList.getLength(); i++) {
               Node nNode = nodeList.item(i);
               //Querying the Elements
               if(nNode.getTextContent().equals("Bentley 2"))
                  found=true;
            }
            if(found)
               System.out.println("Bentley 2 car is found");
            else
                System.out.println("Bentley 2 car is not found");
         } catch (Exception e) {
            e.printStackTrace();
         }
      }
   }
```

Output

The output window displays that the Bentley 2 car is found.

```
Bentley 2 car is found
```


Querying by Attributes

To query attributes inside elements, we use the XPath expression **'/root_element/element_name/@attr_name'**. This expression fetches all the

attributes with specified name of the specified element inside the root element. All these attributes are in the form of nodes inside the NodeList.

Example 1

We use the same **cars.xml** file we have used in the previous example to count the total number of Bentley cars. We have used the expression, **'/cars/carname/@company'** to get all the company attribute nodes and by checking it with 'Bentley', we are incrementing the count variable.

```
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;

public class QueryByAttributes {
   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("cars.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 = "/cars/carname/@company";
         NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(
            doc, XPathConstants.NODESET);
```

```
         //Iterating over NodeList
         int count=0;
         for (int i = 0; i < nodeList.getLength(); i++) {
            Node nNode = nodeList.item(i);
            //Querying the Elements
            if(nNode.getNodeValue().equals("Bentley"))
               count++;
         }
         System.out.println("Number of Bentley cars found: " + count);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Output

The output window displays number of Bentley cars found in the XML document.

```
Number of Bentley cars found: 3
```

Example 2

We need to query the following **studentData.xml** to get only the information related to the student with roll number 493.

```
<?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>singn</lastname>
      <nickname>jazz</nickname>
      <marks>90</marks>
   </student>
</class>
```

In the following **QueryXPathDemo.java** program, we have parsed the above studentData.xml file and built a document. The XPath expression, **'/class/student[@rollno = '493']'** is used to get that student element with roll number 493.

```
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 QueryXPathDemo {

   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[@rollno = '493']";
        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());
           //Getting student info with roll number 493
           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();
     }
```

```
    }
}
```

The output window displays the information of the student with roll number 493.

Output

```
Current Element :student
Student roll no : 493
First Name : Vaneet
Last Name : Gupta
Nick Name : vinni
Marks : 95
```