Java StAX Parser - Parse XML Document

The Java StAX parser API has classes, methods and interfaces to parse XML documents in the form of events. It is a pull based API that gives the client program more privilege to access the events only if required. In this chapter, we are going to see how to parse an XML documents in Java using StAX parser API in detail.

Parse XML Using Java StAX Parser

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

- **Step 1:** Creating XMLInputFactory instance
- **Step 2:** Reading the XML
- **Step 3:** Parsing the XML
- **Step 4:** Retrieving the Elements

Step 1: Creating XMLInputFactory instance

The XMLInputFactory class is an abstract class that is used to get input streams. To create a new instance of an XMLInputFactory, we use newInstance() method. If the instance of this factory cannot be loaded, it throws an error named, "FactoryConfigurationError".

```
XMLInputFactory factory = XMLInputFactory.newInstance();
```

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: Parsing the XML

To parse XML events, we create XMLEventReader from the XMLInputFactory object by passing either the FileReader object or the input stream object. If the creation of XMLEventReader is not successful, it throws XMLStreamException.

```
XMLEventReader eventReader = factory.createXMLEventReader(input);
```

Step 4: Retrieving the Elements

The **nextEvent()** method of XMLEventReader returns the next XML event in the form of XMLEvent object. The XMLEvent has methods to return events as startElement, endElement and Characters.

```
XMLEvent event = eventReader.nextEvent();
```

Retrieving Element Name

To retrieve Element name, we should first get the Element from the XML document. When the event is of type XMLStreamConstants.START_ELEMENT, the **asStartElement()** on an XMLEvent object, retrieves the Element in the form of a StartElement object.

The **getName()** method of StartElement returns the name of the Element in the form of a String.

Example

The **RetrieveElementName.java** program takes the XML content in a StringBuilder object and convert it into bytes. The obtained InputStream is used to create XMLEventReader. The Element is accessed using the events notified by the parser.

</> Open Compiler

```
import java.io.ByteArrayInputStream;
import javax.xml.stream.XMLEventReader;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamConstants;
```

```
import javax.xml.stream.events.StartElement;
import javax.xml.stream.events.XMLEvent;

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

         //Creating XMLInputFactory instance
         XMLInputFactory factory = XMLInputFactory.newInstance();

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

         //Parsing the XML
         XMLEventReader eventReader =
         factory.createXMLEventReader(input);

         //Retrieving the Elements
         while(eventReader.hasNext()) {
            XMLEvent event = eventReader.nextEvent();
            if(event.getEventType()==XMLStreamConstants.START_ELEMENT) {
            StartElement startElement = event.asStartElement();
            System.out.println("Element Name: " + startElement.getName());
            }
         }
      } catch(Exception e) {
          e.printStackTrace();
      }
   }
}
```

Output

The name of the Element is displayed on the output screen.

```
Element Name: class
```

Retrieving Text Content

To retrieve text content of an element, **asCharacters()** method is used on XMLEvent object. When the event is of type XMLStreamConstants.CHARACTERS, only then we can use asCharacters() method. This method returns the data in the of Characters object. The **getData()** method is used to get the text content in the form of a String.

Example

In the previous example, we have taken XML content as an Input Stream. Now, let us take input by reading from a file by saving the following XML content in a file named, **classData.xml**

```
<class>xyz</class>
```

In the following **RetrievingTextContent.java** program, we have read the classData.xml file using a FileReader object and passed as an input to XMLEventReader. Using, XMLEvent object, we have obtained the text content of the Element.

```
import java.io.FileReader;
import javax.xml.stream.XMLEventReader;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamConstants;
import javax.xml.stream.events.Characters;
import javax.xml.stream.events.XMLEvent;

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

         //Creating XMLInputFactory instance
         XMLInputFactory factory = XMLInputFactory.newInstance();

         //Reading the XML
         FileReader fileReader = new FileReader("classData.xml");

         //Parsing the XML
         XMLEventReader eventReader =
         factory.createXMLEventReader(fileReader);
```

```
         //Retrieving the Elements
         while(eventReader.hasNext()) {
            XMLEvent event = eventReader.nextEvent();
            if(event.getEventType()==XMLStreamConstants.CHARACTERS) {
                Characters characters = event.asCharacters();
                System.out.println("Text Content : "+ characters.getData());
            }
         }
      } catch(Exception e) {
          e.printStackTrace();
      }
   }
}
```

Output

The text content of the element is displayed on the output screen.

```
Text Content : xyz
```

Retrieving Attributes

The **getAttributes()** method of StartElement interface returns a readonly Iterator of attributes declared on this element. If there are no attributes declared on this element, it returns an empty iterator.

The **getValue()** function on Attribute interface returns the value of the attribute in the form of a String.

Example

The following **classData.xml** has the information of three students along with their roll numbers as attributes. Let us retrieve this information using StAX API in Java.

```
<?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 **RetrieveAttributes.java** program, we have used switch case statements for START_ELEMENT, CHARACTERS and END_ELEMENT XMLStreamConstants to access all the information of elements.

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Iterator;
import javax.xml.stream.XMLEventReader;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamConstants;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.events.Attribute;
import javax.xml.stream.events.Characters;
import javax.xml.stream.events.EndElement;
import javax.xml.stream.events.StartElement;
import javax.xml.stream.events.XMLEvent;

public class RetrievingAttributes {
   public static void main(String[] args) {
      boolean bFirstName = false;
      boolean bLastName = false;
      boolean bNickName = false;
      boolean bMarks = false;

      try {
```

```
        //Creating XMLInputFactory instance
        XMLInputFactory factory = XMLInputFactory.newInstance();

        //Reading the XML
        FileReader fileReader = new FileReader("classData.xml");

        //Parsing the XML
        XMLEventReader eventReader =
        factory.createXMLEventReader(fileReader);

        //Retrieving the Elements
        while(eventReader.hasNext()) {
           XMLEvent event = eventReader.nextEvent();

           switch(event.getEventType()) {

              case XMLStreamConstants.START_ELEMENT:
                 StartElement startElement = event.asStartElement();
                 String qName = startElement.getName().getLocalPart();

              if (qName.equalsIgnoreCase("student")) {
                 System.out.println("Start Element : student");
                 Iterator<Attribute> attributes =
startElement.getAttributes();
                 String rollNo = attributes.next().getValue();
                 System.out.println("Roll No : " + rollNo);
              } else if (qName.equalsIgnoreCase("firstname")) {
                 bFirstName = true;
              } else if (qName.equalsIgnoreCase("lastname")) {
                 bLastName = true;
              } else if (qName.equalsIgnoreCase("nickname")) {
                 bNickName = true;
              }
              else if (qName.equalsIgnoreCase("marks")) {
                 bMarks = true;
              }
              break;

              case XMLStreamConstants.CHARACTERS:
                 Characters characters = event.asCharacters();
              if(bFirstName) {
                 System.out.println("First Name: " + characters.getData());
```

```
                bFirstName = false;
            }
            if(bLastName) {
                System.out.println("Last Name: " + characters.getData());
                bLastName = false;
            }
            if(bNickName) {
                System.out.println("Nick Name: " + characters.getData());
                bNickName = false;
            }
            if(bMarks) {
                System.out.println("Marks: " + characters.getData());
                bMarks = false;
            }
            break;

            case XMLStreamConstants.END_ELEMENT:
                EndElement endElement = event.asEndElement();


if(endElement.getName().getLocalPart().equalsIgnoreCase("student")) {
                System.out.println("End Element : student");
                System.out.println();
            }
            break;
         }
      }
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (XMLStreamException e) {
         e.printStackTrace();
      }
   }
}
```

Output

All the information of students along with their roll numbers are displayed on the output screen.

```
Start Element : student
Roll No : 393
```

```
First Name: dinkar
Last Name: kad
Nick Name: dinkar
Marks: 85
End Element : student

Start Element : student
Roll No : 493
First Name: Vaneet
Last Name: Gupta
Nick Name: vinni
Marks: 95
End Element : student

Start Element : student
Roll No : 593
First Name: jasvir
Last Name: singn
Nick Name: jazz
Marks: 90
End Element : student
```