Java StAX Parser - Modify XML Document

Java StAX Parser is a java API which provides interfaces, classes and methods to parse and modify XML documents. Random access of XML elements is not possible using StAX parser and hence, we need to store the required data as the parser parses the XML document. We can use XMLEventReader to read the XML document and XMLEventWriter to write the updated content simultaneously.

Modify XML Using Java StAX Parser

We can modify an XML document in Java using StAX parser through following steps –

- **Step 1:** Creating XMLInputFactory instance
- **Step 2:** Reading the XML
- **Step 3:** Parsing the XML
- **Step 4:** Modifying the XML content
- **Step 5:** Creating XMLStreamReader object
- **Step 6:** writing the updated content into XML file

Refer Parse XML Document of this section for first three steps

Step 4: Modifying the XML content

After following the first three steps, we have our XML file in XMLEventReader object. Since, we don't have random access of XML elements, we can read and write the XML content simultaneously. Here, updating means simply creating new events and adding them in the place of old events.

To create any XMLEvent, we should first get the instance of XMLEventFactory. Using XMLEventFactory, we can add new elements and attributes.

```
XMLEventFactory eventFactory=XMLEventFactory.newInstance();
```

Refer Create XML Document chapter of this section for steps 5 and 6.

Updating Text Content

The **createCharacters("text")** method of XMLEventFactory creates character content with the text specified and returns Characters event. This event can be added to the XMLEventWriter using add() function.

Example

Here is the **studentData.xml** file in which we need to update the marks for the student with roll number 593.

```
<?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 **UpdatingTextContent.java** program, we have used two boolean variables studentFound and marksFound to identify that student with the roll number 593. We are adding all the events to XMLEventWriter except the Characters event of marks for student with roll number 593. We are creating a new Characters event for that student and adding it to the XMLEventWriter.

```
import java.io.FileReader;
import java.io.StringWriter;
import javax.xml.namespace.QName;
import javax.xml.stream.XMLEventFactory;
import javax.xml.stream.XMLEventReader;
import javax.xml.stream.XMLEventWriter;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamConstants;
```

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

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

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

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

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

         //Updating text content
         StringWriter stringWriter = new StringWriter();
         XMLOutputFactory outFactory = XMLOutputFactory.newInstance();
         XMLEventWriter eventWriter =
outFactory.createXMLEventWriter(stringWriter);
         XMLEventFactory eventFactory=XMLEventFactory.newInstance();

         boolean studentFound=false;
         boolean marksFound=false;
         while(eventReader.hasNext()) {
            XMLEvent event = eventReader.nextEvent();
            if(event.getEventType()==XMLStreamConstants.START_ELEMENT) {
               StartElement startElement = event.asStartElement();
               String qName = startElement.getName().getLocalPart();
               Attribute attr = startElement.getAttributeByName(new
QName("rollno"));

               if (qName.equalsIgnoreCase("student") &&
attr.getValue().equals("593")) {
                  studentFound=true;
                  }
               if (qName.equalsIgnoreCase("marks") && studentFound) {
                  studentFound = false;
                  marksFound=true;
```

```
               }
            }
            if(event.getEventType()==XMLStreamConstants.CHARACTERS &&
marksFound) {
               eventWriter.add(eventFactory.createCharacters("64"));
               marksFound=false;
            }
            else {
               eventWriter.add(event);
            }
         }
         String xmlString = stringWriter.getBuffer().toString();
         System.out.println(xmlString);

      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

The output window displays the student data with the marks modified for student with roll number 593.

```
<?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>64</marks>
```

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

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

Adding Elements to Existing XML File

The **add(XMLEvent event)** function of XMLEventWriter adds the event to the Writer or the OutputStream specified while creating XMLEventWriter. Adding a new StartElement event opens the new namespace scope and will be closed when an EndElement event is added.

Using the same studentData.xml file that we have discussed in the previous example, we are going to add new student element with all the necessary information.

Example

In the following **AddXMLElements.java** program, we have added all the events to the XMLEventWriter till the last student element. We have added the new student element at the end and then closed the root element. To find the exact position of adding the new element, we used peek() method, since it just shows the peek event instead of reading it from the stream.

```
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileReader;
import java.io.StringWriter;
import javax.xml.stream.XMLEventFactory;
import javax.xml.stream.XMLEventReader;
import javax.xml.stream.XMLEventWriter;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamConstants;
import javax.xml.stream.XMLStreamReader;
import javax.xml.stream.events.XMLEvent;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stax.StAXSource;
import javax.xml.transform.stream.StreamResult;

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

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

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

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

         //Modifying the XML content
         StringWriter stringWriter = new StringWriter();
         XMLOutputFactory outFactory = XMLOutputFactory.newInstance();
         XMLEventWriter eventWriter =
outFactory.createXMLEventWriter(stringWriter);
         XMLEventFactory eventFactory=XMLEventFactory.newInstance();

         while(eventReader.hasNext()) {
            XMLEvent event = eventReader.nextEvent();
            if(event.getEventType()==XMLStreamConstants.END_ELEMENT &&

eventReader.peek().getEventType()==XMLStreamConstants.END_DOCUMENT ) {

               eventWriter.add(eventFactory.createStartElement("", "",
"student"));
               eventWriter.add(eventFactory.createAttribute("rollno", "693"));

               eventWriter.add(eventFactory.createStartElement("", "",
"firstname"));
               eventWriter.add(eventFactory.createCharacters("Daniel"));
               eventWriter.add(eventFactory.createEndElement("", "",
"firstname"));

               eventWriter.add(eventFactory.createStartElement("", "",
"lastname"));
               eventWriter.add(eventFactory.createCharacters("Wesley"));
               eventWriter.add(eventFactory.createEndElement("", "",
"lastname"));

               eventWriter.add(eventFactory.createStartElement("", "",
"nickname"));
```

```
               eventWriter.add(eventFactory.createCharacters("Dany"));
               eventWriter.add(eventFactory.createEndElement("", "",
"nickname"));

               eventWriter.add(eventFactory.createStartElement("", "",
"marks"));
               eventWriter.add(eventFactory.createCharacters("75"));
               eventWriter.add(eventFactory.createEndElement("", "",
"marks"));

               eventWriter.add(eventFactory.createEndElement("", "",
"student"));
            }
            else {
               eventWriter.add(event);
            }
         }

         //Creating XMLStreamReader object
         String xmlString = stringWriter.getBuffer().toString();
         ByteArrayInputStream input = new
ByteArrayInputStream(xmlString.getBytes("UTF-8"));
         stringWriter.close();
         XMLStreamReader streamReader =
                 factory.createXMLStreamReader(input);

         //writing the updated content into XML file
         TransformerFactory transformerFactory =
TransformerFactory.newInstance();
         Transformer transformer = transformerFactory.newTransformer();
         StAXSource source = new StAXSource(streamReader);
         StreamResult result = new StreamResult(new File("studentData.xml"));
         transformer.transform(source, result);

         System.out.println(xmlString);

      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Output

This is the content of the output file after adding new student information−

```
<?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>
 <student rollno="693">
   <firstname>Daniel</firstname>
   <lastname>Wesley</lastname>
   <nickname>Dany</nickname>
   <marks>75</marks>
 </student>
</class>
```