APACHE XERCES - QUICK GUIDE

XML OVERVIEW

What is XML?

XML is a simple text based language which was designed to store and transport data in plain text format. It stands for Extensible Markup Language. Following are some of the salient features of XML.

- XML is a markup language.
- XML is a tag based language like HTML.
- XML tags are not predefined like HTML.
- You can define your own tags which is why it is called extensible language.
- XML tags are designed to be self descriptive.
- XML is W3C Recommendation for data storage and transport.

Example

```
<?xml version="1.0"?>
<Class>
   <Name>First</Name>
   <Sections>
      <Section>
         <Name>A</Name>
         <Students>
            <Student>Rohan</Student>
            <Student>Mohan</Student>
            <Student>Sohan</Student>
            <Student>Lalit</Student>
            <Student>Vinay</Student>
         </Students>
      </Section>
      <Section>
         <Name>B</Name>
         <Students>
            <Student>Robert</Student>
            <Student>Julie</Student>
            <Student>Kalie</Student>
            <Student>Michael</Student>
         </Students>
      </Section>
   </Sections>
</Class>
```

Advantages

Following are the advantages that XML provides:

- **Technology agnostic** - Being plain text, XML is technology independent. It can be used by any technology for data storage and transmission purpose.
- **Human readable**- XML uses simple text format. It is human readable and understandable.
- **Extensible** - in XML, custom tags can be created and used very easily.
- **Allow Validation** - Using XSD, DTD and XML structure can be validated easily.

Disadvantages

Following are the disadvantages of XML usage:

- **Redundant Syntax** - Normally XML file contains lot of repeatitive terms.
- **Verbose**-Being a verbose language, XML file size increases the transmission and storage costs.

APACHE XERCES - ENVIRONMENT SETUP

This chapter takes you through the process of setting up Apache Xerces on Windows and Linux based systems. Apache Xerces can be easily installed and integrated with your current Java environment following a few simple steps without any complex setup procedures. User administration is required while installation.

System Requirements

JDK	Java SE 2 JDK 1.5 or above
Memory	1 GB RAM (recommended)
Disk Space	No minimum requirement
Operating System Version	Windows XP or above, Linux

Let us now proceed with the steps to install Apache Xerces.

Step 1: Verify your Java Installation

First of all, you need to have Java Software Development Kit (SDK) installed on your system. To verify this, execute any of the two commands depending on the platform you are working on.

If the Java installation has been done properly, then it will display the current version and specification of your Java installation. A sample output is given in the following table.

Platform	Command	Sample Output
Windows	Open Command Console and type: java -version	Java version "1.7.0_60" Java (TM) SE Run Time Environment (build 1.7.0_60-b19) Java Hotspot (TM) 64-bit Server VM (build 24.60-b09,mixed mode)
Linux	Open command terminal and type: $java -version	java version "1.7.0_25" Open JDK Runtime Environment (rhel-2.3.10.4.el6_4-x86_64) Open JDK 64-Bit Server VM (build 23.7-b01, mixed mode)

- We assume the readers of this tutorial have Java SDK version 1.7.0_60 installed on their system.
- In case you do not have Java SDK, download its current version from http://www.oracle.com/technetwork/java/javase/downloads/index.html and have it installed.

Step 2: Set your Java Environment

Set the environment variable JAVA_HOME to point to the base directory location where Java is installed on your machine. For example,

Platform	Description
Windows	Set JAVA_HOME to C:\ProgramFiles\java\jdk1.7.0_60
Linux	Export JAVA_HOME=/usr/local/java-current

Append the full path of Java compiler location to the System Path.

Platform	Description
Windows	Linux
Append the String "C:\Program Files\Java\jdk1.7.0_60\bin" to the end of the system variable PATH.	Export PATH=$PATH:$JAVA_HOME/bin/

Execute the command java -version from the command prompt as explained above.

Step 3: Install Apache Xerces Library

Download the latest version of Apache Xerces from http://xcerces.apache.org/download.html and unzip its contents to a folder from where the required libraries can be linked to your Java program. Let us assume the files are collected in a folder xerces-2_11_0 on C drive.

Add the complete path of the five jars as highlighted in the above image to the CLASSPATH.

Platform	Description
Windows	Append the following strings to the end of the user variable CLASSPATH: C:\xerces-2_11_0\resolver.jar; C:\xerces-2_11_0\serializer.jar; C:\xerces-2_11_0\xercesImpl.jar; C:\xerces-2_11_0\xercesSamples.jar; C:\xerces-2_11_0\xml-apis.jar;
Linux	Export CLASSPATH=$CLASSPATH: /usr/share/xerces-2_11_0\resolver.jar: /usr/share/xerces-2_11_0\serializer.jar: /usr/share/xerces-2_11_0\xercesImpl.jar: /usr/share/xerces-2_11_0\xercesSamples.jar: /usr/share/xerces-2_11_0\xml-apis.jar:

APACHE XERCES XML PARSERS

What is Apache Xerces2?

Xerces2 is a java based processor and provides standard interfaces and implementations for following XML parsing API standards:

- Document Object Model (DOM) Level 3
- Simple API for XML (SAX) 2.0.2
- Streaming API For XML (StAX) 1.0 Event API
- Java APIs for XML Processing (JAXP) 1.4

What is XML Parsing?

Parsing XML refers to going through XML document to access data or to modify data in one or other way.

What is XML Parser?

XML Parser provides way how to access or modify data present in an XML document. Java provides multiple options to parse XML document. Following are various types of parsers which are commonly used to parse XML documents.

- **Dom Parser** - Parses the document by loading the complete contents of the document and creating its complete hiearchical tree in memory.
- **SAX Parser** - Parses the document on event based triggers. Does not load the complete document into the memory.
- **StAX Parser** - Parses the document in similar fashion to SAX parser but in more efficient way.

We'll elboborate each parser using Apache Xerces library in detail in next chapters.

APACHE XERCES DOM PARSER - OVERVIEW

The Document Object Model is an official recommendation of the World Wide Web Consortium (W3C). It defines an interface that enables programs to access and update the style, structure,and contents of XML documents. XML parsers that support the DOM implement that interface.

When to use?

You should use a DOM parser when:

- You need to know a lot about the structure of a document
- You need to move parts of the document around (you might want to sort certain elements, for example)
- You need to use the information in the document more than once

What you get?

When you parse an XML document with a DOM parser, you get back a tree structure that contains all of the elements of your document. The DOM provides a variety of functions you can use to examine the contents and structure of the document.

Advantages

The DOM is a common interface for manipulating document structures. One of its design goals is

that Java code written for one DOM-compliant parser should run on any other DOM-compliant parser without changes.

DOM interfaces

The DOM defines several Java interfaces. Here are the most common interfaces:

- **Node** - The base datatype of the DOM.
- **Element** - The vast majority of the objects you'll deal with are Elements.
- **Attr** Represents an attribute of an element.
- **Text** The actual content of an Element or Attr.
- **Document** Represents the entire XML document. A Document object is often referred to as a DOM tree.

Common DOM methods

When you are working with the DOM, there are several methods you'll use often:

- **Document.getDocumentElement()** - Returns the root element of the document.
- **Node.getFirstChild()** - Returns the first child of a given Node.
- **Node.getLastChild()** - Returns the last child of a given Node.
- **Node.getNextSibling()** - These methods return the next sibling of a given Node.
- **Node.getPreviousSibling()** - These methods return the previous sibling of a given Node.
- **Node.getAttribute(attrName)** - For a given Node, returns the attribute with the requested name.

APACHE XERCES DOM PARSER - PARSE XML DOCUMENT

Steps to Using DOM

Following are the steps used while parsing a document using DOM Parser.

- Import XML-related packages.
- Create a DocumentBuilder
- Create a Document from a file or stream
- Extract the root element
- Examine attributes
- Examine sub-elements

Import XML-related packages

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;
```

Create a DocumentBuilder

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

Create a Document from a file or stream

```
StringBuilder xmlStringBuilder = new StringBuilder();
xmlStringBuilder.append("<?xml version="1.0"?> <class> </class>");
ByteArrayInputStream input =  new ByteArrayInputStream(
   xmlStringBuilder.toString().getBytes("UTF-8"));
Document doc = builder.parse(input);
```

Extract the root element

```
Element root = document.getDocumentElement();
```

Examine attributes

```
//returns specific attribute
getAttribute("attributeName");
//returns a Map (table) of names/values
getAttributes();
```

Examine sub-elements

```
//returns a list of subelements of specified name
getElementsByTagName("subelementName");
//returns a list of all child nodes
getChildNodes();
```

Demo Example

Here is the input xml file we need to parse:

```
<?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>Vineet</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>
```

Demo Example:

DomParserDemo.java

```
package com.tutorialspoint.xml;

import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;

public class DomParserDemo {
```

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

      try {
         File inputFile = new File("input.txt");
         DocumentBuilderFactory dbFactory
            = DocumentBuilderFactory.newInstance();
         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
         Document doc = dBuilder.parse(inputFile);
         doc.getDocumentElement().normalize();
         System.out.println("Root element :"
            + doc.getDocumentElement().getNodeName());
         NodeList nList = doc.getElementsByTagName("student");
         System.out.println("----------------------------");
         for (int temp = 0; temp < nList.getLength(); temp++) {
            Node nNode = nList.item(temp);
            System.out.println("\nCurrent Element :"
               + nNode.getNodeName());
            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();
      }
   }
}
```

This would produce the following result:

```
Root element :class
----------------------------

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 : Vineet
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
```

APACHE XERCES DOM PARSER - QUERY XML DOCUMENT

Demo Example

Here is the input xml file we need to query:

```
<?xml version="1.0"?>
<cars>
   <supercars company="Ferrari">
      <carname type="formula one">Ferarri 101</carname>
      <carname type="sports car">Ferarri 201</carname>
      <carname type="sports car">Ferarri 301</carname>
   </supercars>
   <supercars company="Lamborgini">
      <carname>Lamborgini 001</carname>
      <carname>Lamborgini 002</carname>
      <carname>Lamborgini 003</carname>
   </supercars>
   <luxurycars company="Benteley">
      <carname>Benteley 1</carname>
      <carname>Benteley 2</carname>
      <carname>Benteley 3</carname>
   </luxurycars>
</cars>
```

Demo Example:

QueryXmlFileDemo.java

```
package com.tutorialspoint.xml;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;
import java.io.File;

public class QueryXmlFileDemo {

   public static void main(String argv[]) {

      try {
         File inputFile = new File("input.txt");
         DocumentBuilderFactory dbFactory =
            DocumentBuilderFactory.newInstance();
         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
         Document doc = dBuilder.parse(inputFile);
         doc.getDocumentElement().normalize();
         System.out.print("Root element: ");
         System.out.println(doc.getDocumentElement().getNodeName());
         NodeList nList = doc.getElementsByTagName("supercars");
         System.out.println("----------------------------");
         for (int temp = 0; temp < nList.getLength(); temp++) {
            Node nNode = nList.item(temp);
            System.out.println("\nCurrent Element :");
            System.out.print(nNode.getNodeName());
            if (nNode.getNodeType() == Node.ELEMENT_NODE) {
               Element eElement = (Element) nNode
               System.out.print("company : ");
               System.out.println(eElement.getAttribute("company"));
```

```
                NodeList carNameList =
                   eElement.getElementsByTagName("carname");
                for (int count = 0;
                   count < carNameList.getLength(); count++) {
                   Node node1 = carNameList.item(count);
                   if (node1.getNodeType() ==
                      node1.ELEMENT_NODE) {
                      Element car = (Element) node1;
                      System.out.print("car name : ");
                      System.out.println(car.getTextContent());
                      System.out.print("car type : ");
                      System.out.println(car.getAttribute("type"));
                   }
                }
             }
          }
       } catch (Exception e) {
          e.printStackTrace();
       }
   }
}
```

This would produce the following result:

```
Root element :cars
----------------------------

Current Element :supercars
company : Ferrari
car name : Ferarri 101
car type : formula one
car name : Ferarri 201
car type : sports car
car name : Ferarri 301
car type : sports car

Current Element :supercars
company : Lamborgini
car name : Lamborgini 001
car type :
car name : Lamborgini 002
car type :
car name : Lamborgini 003
car type :
```

APACHE XERCES DOM PARSER - CREATE XML DOCUMENT

Demo Example

Here is the XML we need to create:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<cars><supercars company="Ferrari">
   <carname type="formula one">Ferrari 101</carname>
   <carname type="sports">Ferrari 202</carname>
</supercars></cars>
```

Demo Example:

CreateXmlFileDemo.java

```
package com.tutorialspoint.xml;

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 CreateXmlFileDemo {

   public static void main(String argv[]) {

      try {
         DocumentBuilderFactory dbFactory =
         DocumentBuilderFactory.newInstance();
         DocumentBuilder dBuilder =
            dbFactory.newDocumentBuilder();
         Document doc = dBuilder.newDocument();
         // root element
         Element rootElement = doc.createElement("cars");
         doc.appendChild(rootElement);

         //  supercars element
         Element supercar = doc.createElement("supercars");
         rootElement.appendChild(supercar);

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

         // carname 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);

         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);

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

This would produce the following result:

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

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

APACHE XERCES DOM PARSER - MODIFY XML DOCUMENT

Demo Example

Here is the input xml file we need to modify:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<cars>
   <supercars company="Ferrari">
      <carname type="formula one">Ferrari 101</carname>
      <carname type="sports">Ferrari 202</carname>
   </supercars>
   <luxurycars company="Benteley">
      <carname>Benteley 1</carname>
      <carname>Benteley 2</carname>
      <carname>Benteley 3</carname>
   </luxurycars>
</cars>
```

Demo Example:

ModifyXmlFileDemo.java

```
package com.tutorialspoint.xml;

import java.io.File;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
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 org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class ModifyXmlFileDemo {

public static void main(String argv[]) {

   try {
      File inputFile = new File("input.txt");
      DocumentBuilderFactory docFactory =
      DocumentBuilderFactory.newInstance();
      DocumentBuilder docBuilder =
      docFactory.newDocumentBuilder();
      Document doc = docBuilder.parse(inputFile);
      Node cars = doc.getFirstChild();
      Node supercar = doc.getElementsByTagName("supercars").item(0);
      // update supercar attribute
      NamedNodeMap attr = supercar.getAttributes();
      Node nodeAttr = attr.getNamedItem("company");
      nodeAttr.setTextContent("Lamborigini");

      // loop the supercar child node
      NodeList list = supercar.getChildNodes();
      for (int temp = 0; temp < list.getLength(); temp++) {
         Node node = list.item(temp);
         if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            if ("carname".equals(eElement.getNodeName())){
```

```
               if("Ferrari 101".equals(eElement.getTextContent())){
                  eElement.setTextContent("Lamborigini 001");
               }
            if("Ferrari 202".equals(eElement.getTextContent()))
               eElement.setTextContent("Lamborigini 002");
            }
         }
      }
      NodeList childNodes = cars.getChildNodes();
      for(int count = 0; count < childNodes.getLength(); count++){
         Node node = childNodes.item(count);
         if("luxurycars".equals(node.getNodeName()))
            cars.removeChild(node);
         }
         // write the content on console
         TransformerFactory transformerFactory =
         TransformerFactory.newInstance();
         Transformer transformer = transformerFactory.newTransformer();
         DOMSource source = new DOMSource(doc);
         System.out.println("-----------Modified File-----------");
         StreamResult consoleResult = new StreamResult(System.out);
         transformer.transform(source, consoleResult);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

This would produce the following result:

```
-----------Modified File-----------
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<cars>
<supercars company="Lamborigini">
<carname type="formula one">Lamborigini 001</carname>
<carname type="sports">Lamborigini 002</carname>
</supercars></cars>
```

APACHE XERCES SAX PARSER - OVERVIEW

SAX (the Simple API for XML) is an event-based parser for xml documents.Unlike a DOM parser, a SAX parser creates no parse tree. SAX is a streaming interface for XML, which means that applications using SAX receive event notifications about the XML document being processed an element, and attribute, at a time in sequential order starting at the top of the document, and ending with the closing of the ROOT element.

- Reads an XML document from top to bottom, recognizing the tokens that make up a well-formed XML document
- Tokens are processed in the same order that they appear in the document
- Reports the application program the nature of tokens that the parser has encountered as they occur
- The application program provides an "event" handler that must be registered with the parser
- As the tokens are identified, callback methods in the handler are invoked with the relevant information

When to use?

You should use a SAX parser when:

- You can process the XML document in a linear fashion from the top down
- The document is not deeply nested

- You are processing a very large XML document whose DOM tree would consume too much memory.Typical DOM implementations use ten bytes of memory to represent one byte of XML
- The problem to be solved involves only part of the XML document
- Data is available as soon as it is seen by the parser, so SAX works well for an XML document that arrives over a stream

Disadvantages of SAX

- We have no random access to an XML document since it is processed in a forward-only manner
- If you need to keep track of data the parser has seen or change the order of items, you must write the code and store the data on your own

ContentHandler Interface

This interface specifies the callback methods that the SAX parser uses to notify an application program of the components of the XML document that it has seen.

- **void startDocument()** - Called at the beginning of a document.
- **void endDocument()** - Called at the end of a document.
- **void startElement(String uri, String localName, String qName, Attributes atts)** - Called at the beginning of an element.
- **void endElement(String uri, String localName,String qName)** - Called at the end of an element.
- **void characters(char[] ch, int start, int length)** - Called when character data is encountered.
- **void ignorableWhitespace(char[] ch, int start, int length)** - Called when a DTD is present and ignorable whitespace is encountered.
- **void processingInstruction(String target, String data)** - Called when a processing instruction is recognized.
- **void setDocumentLocator(Locator locator))** - Provides a Locator that can be used to identify positions in the document.
- **void skippedEntity(String name)** - Called when an unresolved entity is encountered.
- **void startPrefixMapping(String prefix, String uri)** - Called when a new namespace mapping is defined.
- **void endPrefixMapping(String prefix)** - Called when a namespace definition ends its scope.

Attributes Interface

This interface specifies methods for processing the attributes connected to an element.

- **int getLength()** - Returns number of attributes.
- **String getQName(int index)**
- **String getValue(int index)**
- **String getValue(String qname)**

Demo Example

Here is the input xml file we need to parse:

```
<?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>Vineet</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>
```

UserHandler.java

```
package com.tutorialspoint.xml;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class UserHandler extends DefaultHandler {

   boolean bFirstName = false;
   boolean bLastName = false;
   boolean bNickName = false;
   boolean bMarks = false;

   @Override
   public void startElement(String uri,
   String localName, String qName, Attributes attributes)
      throws SAXException {
      if (qName.equalsIgnoreCase("student")) {
         String rollNo = attributes.getValue("rollno");
         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;
      }
   }

   @Override
   public void endElement(String uri,
   String localName, String qName) throws SAXException {
      if (qName.equalsIgnoreCase("student")) {
         System.out.println("End Element :" + qName);
      }
   }
```

```
   @Override
   public void characters(char ch[],
      int start, int length) throws SAXException {
      if (bFirstName) {
         System.out.println("First Name: "
            + new String(ch, start, length));
         bFirstName = false;
      } else if (bLastName) {
         System.out.println("Last Name: "
            + new String(ch, start, length));
         bLastName = false;
      } else if (bNickName) {
         System.out.println("Nick Name: "
            + new String(ch, start, length));
         bNickName = false;
      } else if (bMarks) {
         System.out.println("Marks: "
            + new String(ch, start, length));
         bMarks = false;
      }
   }
}
```

SAXParserDemo.java

```
package com.tutorialspoint.xml;

import java.io.File;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

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

      try {
         File inputFile = new File("input.txt");
         SAXParserFactory factory = SAXParserFactory.newInstance();
         SAXParser saxParser = factory.newSAXParser();
         UserHandler userhandler = new UserHandler();
         saxParser.parse(inputFile, userhandler);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}

class UserHandler extends DefaultHandler {

   boolean bFirstName = false;
   boolean bLastName = false;
   boolean bNickName = false;
   boolean bMarks = false;

   @Override
   public void startElement(String uri,
      String localName, String qName, Attributes attributes)
         throws SAXException {
      if (qName.equalsIgnoreCase("student")) {
         String rollNo = attributes.getValue("rollno");
         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;
         }
      }

      @Override
      public void endElement(String uri,
         String localName, String qName) throws SAXException {
         if (qName.equalsIgnoreCase("student")) {
            System.out.println("End Element :" + qName);
         }
      }

      @Override
      public void characters(char ch[],
         int start, int length) throws SAXException {
         if (bFirstName) {
            System.out.println("First Name: "
            + new String(ch, start, length));
            bFirstName = false;
         } else if (bLastName) {
            System.out.println("Last Name: "
            + new String(ch, start, length));
            bLastName = false;
         } else if (bNickName) {
            System.out.println("Nick Name: "
            + new String(ch, start, length));
            bNickName = false;
         } else if (bMarks) {
            System.out.println("Marks: "
            + new String(ch, start, length));
            bMarks = false;
         }
      }
   }
```

This would produce the following result:

```
Roll No : 393
First Name: Dinkar
Last Name: Kad
Nick Name: Dinkar
Marks: 85
End Element :student
Roll No : 493
First Name: Vineet
Last Name: Gupta
Nick Name: Vinni
Marks: 95
End Element :student
Roll No : 593
First Name: Jasvir
Last Name: singh
Nick Name: Jazz
Marks: 90
End Element :student
```

APACHE XERCES SAX PARSER - QUERY XML DOCUMENT

Demo Example

Here is the input text file we need to Query for rollno: 393

```
<?xml version="1.0"?>
&ltclass>
   &ltstudent rollno="393">
```

```
      &ltfirstname>Dinkar</firstname>
      &ltlastname>Kad</lastname>
      &ltnickname>Dinkar</nickname>
      &ltmarks>85</marks>
   </student>
   &ltstudent rollno="493">
      &ltfirstname>Vineet</firstname>
      &ltlastname>Gupta</lastname>
      &ltnickname>Vinni</nickname>
      &ltmarks>95</marks>
   </student>
   &ltstudent rollno="593">
      &ltfirstname>Jasvir</firstname>
      &ltlastname>singh</lastname>
      &ltnickname>Jazz</nickname>
      &ltmarks>90</marks>
   </student>
</class>
```

UserHandler.java

```
package com.tutorialspoint.xml;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;


public class UserHandler extends DefaultHandler {

   boolean bFirstName = false;
   boolean bLastName = false;
   boolean bNickName = false;
   boolean bMarks = false;
   String rollNo = null;

   @Override
   public void startElement(String uri,
   String localName, String qName, Attributes attributes)
   throws SAXException {

      if (qName.equalsIgnoreCase("student")) {
         rollNo = attributes.getValue("rollno");
      }
      if(("393").equals(rollNo) &&
         qName.equalsIgnoreCase("student")){
         System.out.println("Start Element :" + qName);
      }
      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;
      }
   }

   @Override
   public void endElement(String uri,
      String localName, String qName) throws SAXException {
      if (qName.equalsIgnoreCase("student")) {
         if(("393").equals(rollNo)
            && qName.equalsIgnoreCase("student"))
            System.out.println("End Element :" + qName);
      }
   }
```

```
   @Override
   public void characters(char ch[],
      int start, int length) throws SAXException {

      if (bFirstName && ("393").equals(rollNo)) {
         //age element, set Employee age
         System.out.println("First Name: " +
            new String(ch, start, length));
         bFirstName = false;
      } else if (bLastName && ("393").equals(rollNo)) {
         System.out.println("Last Name: " +
         new String(ch, start, length));
         bLastName = false;
      } else if (bNickName && ("393").equals(rollNo)) {
         System.out.println("Nick Name: " +
         new String(ch, start, length));
         bNickName = false;
      } else if (bMarks && ("393").equals(rollNo)) {
         System.out.println("Marks: " +
         new String(ch, start, length));
         bMarks = false;
      }
   }
}
```

SAXQueryDemo.java

```
package com.tutorialspoint.xml;

import java.io.File;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class SAXQueryDemo.java {
   public static void main(String[] args){

      try {
         File inputFile = new File("input.txt");
         SAXParserFactory factory = SAXParserFactory.newInstance();
         SAXParser saxParser = factory.newSAXParser();
         UserHandler userhandler = new UserHandler();
         saxParser.parse(inputFile, userhandler);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}

class UserHandler extends DefaultHandler {

   boolean bFirstName = false;
   boolean bLastName = false;
   boolean bNickName = false;
   boolean bMarks = false;
   String rollNo = null;

   @Override
   public void startElement(String uri,
      String localName, String qName, Attributes attributes)
      throws SAXException {

      if (qName.equalsIgnoreCase("student")) {
         rollNo = attributes.getValue("rollno");
```

```
      }
      if(("393").equals(rollNo) &&
         qName.equalsIgnoreCase("student")){
         System.out.println("Start Element :" + qName);
      }
      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;
      }
   }

   @Override
   public void endElement(String uri,
      String localName, String qName) throws SAXException {
      if (qName.equalsIgnoreCase("student")) {
         if(("393").equals(rollNo)
            && qName.equalsIgnoreCase("student"))
            System.out.println("End Element :" + qName);
         }
      }


   @Override
   public void characters(char ch[],
      int start, int length) throws SAXException {

      if (bFirstName && ("393").equals(rollNo)) {
         //age element, set Employee age
         System.out.println("First Name: " +
         new String(ch, start, length));
         bFirstName = false;
      } else if (bLastName && ("393").equals(rollNo)) {
         System.out.println("Last Name: " +
         new String(ch, start, length));
         bLastName = false;
      } else if (bNickName && ("393").equals(rollNo)) {
         System.out.println("Nick Name: " +
         new String(ch, start, length));
         bNickName = false;
      } else if (bMarks && ("393").equals(rollNo)) {
         System.out.println("Marks: " +
         new String(ch, start, length));
         bMarks = false;
      }
   }
}
```

This would produce the following result:

```
Start Element :student
First Name: Dinkar
Last Name: Kad
Nick Name: Dinkar
Marks: 85
End Element :student
```

APACHE XERCES SAX PARSER - CREATE XML DOCUMENT

It is better to use StAX parser for creating XML than using SAX parser. Please refer the Java StAX Parser section for the same.

Demo Example

Here is the input xml file we need to Modify by appending <Result>Pass<Result/>

at the end of </marks> tag

```
<?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>Vineet</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>
```

SAXModifyDemo.java

```
package com.tutorialspoint.xml;

import java.io.*;
import org.xml.sax.*;
import javax.xml.parsers.*;
import org.xml.sax.helpers.DefaultHandler;

public class SAXModifyDemo extends DefaultHandler {
   static String displayText[] = new String[1000];
   static int numberLines = 0;
   static String indentation = "";

   public static void main(String args[]) {

      try {
         File inputFile = new File("input.txt");
         SAXParserFactory factory =
         SAXParserFactory.newInstance();
         SAXModifyDemo obj = new SAXModifyDemo();
            obj.childLoop(inputFile);
         FileWriter filewriter = new FileWriter("newfile.xml");
         for(int loopIndex = 0; loopIndex < numberLines; loopIndex++){
            filewriter.write(displayText[loopIndex].toCharArray());
            filewriter.write('\n');
            System.out.println(displayText[loopIndex].toString());
         }
         filewriter.close();
      }
      catch (Exception e) {
         e.printStackTrace(System.err);
      }
   }

   public void childLoop(File input){
      DefaultHandler handler = this;
```

```
        SAXParserFactory factory = SAXParserFactory.newInstance();
      try {
        SAXParser saxParser = factory.newSAXParser();
        saxParser.parse(input, handler);
      } catch (Throwable t) {}
   }

   public void startDocument() {
      displayText[numberLines] = indentation;
      displayText[numberLines] += "<?xml version=\"1.0\" encoding=\""+
        "UTF-8" + "\"?>";
      numberLines++;
   }

   public void processingInstruction(String target,
      String data) {
      displayText[numberLines] = indentation;
      displayText[numberLines] += "<?";
      displayText[numberLines] += target;
      if (data != null && data.length() > 0) {
        displayText[numberLines] += ' ';
        displayText[numberLines] += data;
      }
      displayText[numberLines] += "?>";
      numberLines++;
   }

   public void startElement(String uri, String localName,
      String qualifiedName, Attributes attributes) {
      displayText[numberLines] = indentation;

      indentation += "    ";

      displayText[numberLines] += '<';
      displayText[numberLines] += qualifiedName;
      if (attributes != null) {
        int numberAttributes = attributes.getLength();
        for (int loopIndex = 0; loopIndex < numberAttributes;
           loopIndex++){
           displayText[numberLines] += ' ';
           displayText[numberLines] += attributes.getQName(loopIndex);
           displayText[numberLines] += "=\"";
           displayText[numberLines] += attributes.getValue(loopIndex);
           displayText[numberLines] += '"';
        }
      }
      displayText[numberLines] += '>';
      numberLines++;
   }

   public void characters(char characters[],
      int start, int length) {
      String characterData = (new String(characters, start, length)).trim();
      if(characterData.indexOf("\n") < 0 && characterData.length() > 0) {
        displayText[numberLines] = indentation;
        displayText[numberLines] += characterData;
        numberLines++;
      }
   }

   public void endElement(String uri, String localName,
      String qualifiedName) {
      indentation = indentation.substring(0, indentation.length() - 4) ;
      displayText[numberLines] = indentation;
      displayText[numberLines] += "</";
      displayText[numberLines] += qualifiedName;
      displayText[numberLines] += '>';
      numberLines++;
```

```
      if (qualifiedName.equals("marks")) {
            startElement("", "Result", "Result", null);
            characters("Pass".toCharArray(), 0, "Pass".length());
            endElement("", "Result", "Result");
      }
   }
}
```

This would produce the following result:

```
<?xml version="1.0" encoding="UTF-8"?>
<class>
    <student rollno="393">
        <firstname>
            Dinkar
        </firstname>
        <lastname>
            Kad
        </lastname>
        <nickname>
            Dinkar
        </nickname>
        <marks>
            85
        </marks>
        <Result>
            Pass
        </Result>
    </student>
    <student rollno="493">
        <firstname>
            Vineet
        </firstname>
        <lastname>
            Gupta
        </lastname>
        <nickname>
            Vinni
        </nickname>
        <marks>
            95
        </marks>
        <Result>
            Pass
        </Result>
    </student>
    <student rollno="593">
        <firstname>
            Jasvir
        </firstname>
        <lastname>
            singh
        </lastname>
        <nickname>
            Jazz
        </nickname>
        <marks>
            90
        </marks>
        <Result>
            Pass
        </Result>
    </student>
</class>
```

APACHE XERCES STAX PARSER - OVERVIEW

StAX is a JAVA based API to parse XML document in a similar way as SAX parser does. But there

are two major difference between the two APIs

- StAX is a PULL API where as SAX is a PUSH API. It means in case of StAX parser, client application need to ask StAX parser to get information from XML whenever it needs but in case of SAX parser, client application is required to get information when SAX parser notifies the client application that information is available.
- StAX API can read as well as write XML documents. Using SAX API, xml can be only be read.

Following are the features of StAX API

- Reads an XML document from top to bottom, recognizing the tokens that make up a well-formed XML document
- Tokens are processed in the same order that they appear in the document
- Reports the application program the nature of tokens that the parser has encountered as they occur
- The application program provides an "event" reader which acts as an iterator and iterates over the event to get the required information. Another reader available is "cursor" reader which acts as a pointer to xml nodes.
- As the events are identified, xml elements can be retrieved from the event object and can be processed further.

When to use?

You should use a StAX parser when:

- You can process the XML document in a linear fashion from the top down.
- The document is not deeply nested.
- You are processing a very large XML document whose DOM tree would consume too much memory. Typical DOM implementations use ten bytes of memory to represent one byte of XML.
- The problem to be solved involves only part of the XML document.
- Data is available as soon as it is seen by the parser, so StAX works well for an XML document that arrives over a stream.

Disadvantages of SAX

- We have no random access to an XML document since it is processed in a forward-only manner
- If you need to keep track of data the parser has seen or change the order of items, you must write the code and store the data on your own

XMLEventReader Class

This class provide iterator of events which can be used to iterate over events as they occur while parsing the XML document

- **StartElement asStartElement()** - used to retrieve value and attributes of element.
- **EndElement asEndElement()** - called at the end of a element.
- **Characters asCharacters()** - can be used to obtain characters such a CDATA, whitespace etc.

XMLEventWriter Class

This interface specifies methods for creating an event.

- **add(Event event)** - Add event containing elements to XML.

XMLStreamReader Class

This class provide iterator of events which can be used to iterate over events as they occur while parsing the XML document

- **int next()** - used to retrieve next event.
- **boolean hasNext()** - used to check further events exists or not
- **String getText()** - used to get text of an element
- **String getLocalName()** - used to get name of an element

XMLStreamWriter Class

This interface specifies methods for creating an event.

- **writeStartElement(String localName)** - Add start element of given name.
- **writeEndElement(String localName)** - Add end element of given name.
- **writeAttribute(String localName, String value)** - Write attribute to an element.

APACHE XERCES STAX PARSER - PARSE XML DOCUMENT

Demo Example

Here is the input xml file we need to parse:

```
<?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>Vineet</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>
```

StAXParserDemo.java

```
package com.tutorialspoint.xml;

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 StAXParserDemo {
   public static void main(String[] args) {
      boolean bFirstName = false;
      boolean bLastName = false;
      boolean bNickName = false;
      boolean bMarks = false;
      try {
         XMLInputFactory factory = XMLInputFactory.newInstance();
         XMLEventReader eventReader =
         factory.createXMLEventReader(
            new FileReader("input.txt"));

            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();
      }
   }
}
```

This would produce the following result:

```
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: Vineet
Last Name: Gupta
Nick Name: Vinni
Marks: 95
End Element : student

Start Element : student
Roll No : 593
First Name: Jasvir
Last Name: singh
Nick Name: Jazz
Marks: 90
End Element : student
```

APACHE XERCES STAX PARSER - QUERY XML DOCUMENT

Demo Example

Here is the input xml file we need to parse:

```
<?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>Vineet</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>
```

StAXParserDemo.java

```
package com.tutorialspoint.xml;
```

```
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 StAXQueryDemo {
   public static void main(String[] args) {
      boolean bFirstName = false;
      boolean bLastName = false;
      boolean bNickName = false;
      boolean bMarks = false;
      boolean isRequestRollNo = false;
      try {
         XMLInputFactory factory = XMLInputFactory.newInstance();
         XMLEventReader eventReader =
         factory.createXMLEventReader(
            new FileReader("input.txt"));

         String requestedRollNo = "393";
         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")) {
                     Iterator<Attribute> attributes = startElement.getAttributes();
                     String rollNo = attributes.next().getValue();
                     if(rollNo.equalsIgnoreCase(requestedRollNo)){
                        System.out.println("Start Element : student");
                        System.out.println("Roll No : " + rollNo);
                        isRequestRollNo = true;
                     }
                  } 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 && isRequestRollNo){
                     System.out.println("First Name: "
                     + characters.getData());
                     bFirstName = false;
                  }
                  if(bLastName && isRequestRollNo){
                     System.out.println("Last Name: "
                     + characters.getData());
                     bLastName = false;
                  }
                  if(bNickName && isRequestRollNo){
                     System.out.println("Nick Name: "
                     + characters.getData());
```

```
                     bNickName = false;
                  }
                  if(bMarks && isRequestRollNo){
                     System.out.println("Marks: "
                     + characters.getData());
                     bMarks = false;
                  }
                  break;
               case  XMLStreamConstants.END_ELEMENT:
                  EndElement endElement = event.asEndElement();
                  if(endElement.getName().getLocalPart().equalsIgnoreCase("student") &&
isRequestRollNo){
                     System.out.println("End Element : student");
                     System.out.println();
                     isRequestRollNo = false;
                  }
               break;
            }
         }
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (XMLStreamException e) {
         e.printStackTrace();
      }
   }
}
```

This would produce the following result:

```
Start Element : student
Roll No : 393
First Name: Dinkar
Last Name: Kad
Nick Name: Dinkar
Marks: 85
End Element : student
```

APACHE XERCES STAX PARSER - CREATE XML DOCUMENT

Demo Example

Here is the XML we need to create:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<cars><supercars company="Ferrari">
<carname type="formula one">Ferrari 101</carname>
<carname type="sports">Ferrari 202</carname>
</supercars></cars>
```

Demo Example:

StAXCreateXMLDemo.java

```
package com.tutorialspoint.xml;

import java.io.IOException;
import java.io.StringWriter;

import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamWriter;

public class StAXCreateXMLDemo {
   public static void main(String[] args) {
      try {
         StringWriter stringWriter = new StringWriter();
```

```
         XMLOutputFactory xMLOutputFactory = XMLOutputFactory.newInstance();
         XMLStreamWriter xMLStreamWriter =
xMLOutputFactory.createXMLStreamWriter(stringWriter);

         xMLStreamWriter.writeStartDocument();
         xMLStreamWriter.writeStartElement("cars");

         xMLStreamWriter.writeStartElement("supercars");
         xMLStreamWriter.writeAttribute("company", "Ferrari");

         xMLStreamWriter.writeStartElement("carname");
         xMLStreamWriter.writeAttribute("type", "formula one");
         xMLStreamWriter.writeCharacters("Ferrari 101");
         xMLStreamWriter.writeEndElement();

         xMLStreamWriter.writeStartElement("carname");
         xMLStreamWriter.writeAttribute("type", "sports");
         xMLStreamWriter.writeCharacters("Ferrari 202");
         xMLStreamWriter.writeEndElement();

         xMLStreamWriter.writeEndElement();
         xMLStreamWriter.writeEndDocument();

         xMLStreamWriter.flush();
         xMLStreamWriter.close();

         String xmlString = stringWriter.getBuffer().toString();

         stringWriter.close();

         System.out.println(xmlString);

      } catch (XMLStreamException e) {
         e.printStackTrace();
      } catch (IOException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }
   }
}
```

This would produce the following result:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<cars><supercars company="Ferrari">
<carname type="formula one">Ferrari 101</carname>
<carname type="sports">Ferrari 202</carname>
</supercars></cars>
```

APACHE XERCES STAX PARSER - MODIFY XML DOCUMENT

Demo Example

Here is the XML we need to modify:

```
<?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>Vineet</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>
```

Demo Example:

StAXModifyDemo.java

```
package com.tutorialspoint.xml;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Iterator;
import java.util.List;

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

import org.jdom2.Document;
import org.jdom2.Element;
import org.jdom2.JDOMException;
import org.jdom2.input.SAXBuilder;
import org.jdom2.output.Format;
import org.jdom2.output.XMLOutputter;

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

      try {
         XMLInputFactory factory = XMLInputFactory.newInstance();
         XMLEventReader eventReader =
            factory.createXMLEventReader(
               new FileReader("input.txt"));
         SAXBuilder saxBuilder = new SAXBuilder();
         Document document = saxBuilder.build(new File("input.txt"));
         Element rootElement = document.getRootElement();
         List<Element> studentElements = rootElement.getChildren("student");
         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")) {
                  Iterator<Attribute> attributes = startElement.getAttributes();
                  String rollNo = attributes.next().getValue();
                  if(rollNo.equalsIgnoreCase("393")){
                     //get the student with roll no 393
                     for(int i=0;i < studentElements.size();i++){
                        Element studentElement = studentElements.get(i);

if(studentElement.getAttribute("rollno").getValue().equalsIgnoreCase("393")){
                           studentElement.removeChild("marks");
                           studentElement.addContent(new Element("marks").setText("80"));
```

```
                        }
                     }
                  }
               }
               break;
            }
         }
         XMLOutputter xmlOutput = new XMLOutputter();
         // display xml
         xmlOutput.setFormat(Format.getPrettyFormat());
         xmlOutput.output(document, System.out);
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (XMLStreamException e) {
         e.printStackTrace();
      } catch (JDOMException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

This would produce the following result:

```
<student rollno="393">
   <firstname>Dinkar</firstname>
   <lastname>Kad</lastname>
   <nickname>Dinkar</nickname>
   <marks>80</marks>
</student>
<student rollno="493">
   <firstname>Vineet</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>
```