# Java 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.

## Environment Setup

In order to use StAX parser, you should have stax.jar in your application's classpath. Download [**stax-1.2.0.jar.**](http://dist.codehaus.org/stax/jars/stax-1.2.0.jar)

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.

Java 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>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>

*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: 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

# Java 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>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>

*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

# Java 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>