Java 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)**

# Java SAX 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>

*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: Vaneet

Last Name: Gupta

Nick Name: vinni

Marks: 95

End Element :student

Roll No : 593

First Name: jasvir

Last Name: singn

Nick Name: jazz

Marks: 90

End Element :student

# Java SAX Parser - Query XML Document

## Demo Example

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

<?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>

*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 {

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

# Java 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.**

# Java SAX Parser - Modify XML Document

## 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>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>

*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>

Vaneet

</firstname>

<lastname>

Gupta

</lastname>

<nickname>

vinni

</nickname>

<marks>

95

</marks>

<Result>

Pass

</Result>

</student>

<student rollno="593">

<firstname>

jasvir

</firstname>

<lastname>

singn

</lastname>

<nickname>

jazz

</nickname>

<marks>

90

</marks>

<Result>

Pass

</Result>

</student>

</class>