Java SAX Parser - Overview

Java SAX (Simple API for XML) is an event-based parser to parse XML documents. Unlike DOM parser, SAX parser does not create a parse tree. It will not load the entire document into memory, instead, it reads through the XML document and notifies the client program whenever it encounters elements, attributes, text content and other data items in the form of events. These events are handled by the methods implemented inside the Event Handler.

What does SAX Parser do?

A SAX Parser does the following to a client program –

- Reads the XML document from top to bottom and identifies the tokens.
- Processes the tokens in the same order of their appearance.
- Reports the parser about the nature of the tokens.
- Invokes the callback methods in the Event handler based on the identified tokens.

When to Use Java SAX Parser?

You should use a SAX parser when –

- You want to process an XML document in a linear fashion from top to bottom.
- The document is not deeply nested.
- Your XML document is very large.
- The problem to be solved involves only a part of the XML document.
- You have streaming data (data is available as soon as it is seen by the parser).

Advantages

Following are some advantages of Java SAX Parser –

- Consumes less memory
- It is faster than DOM parser. Because, we need not wait for the entire document to get loaded in the memory.
- You can still process the XML documents larger than the system memory.

Disadvantages

Here are some of the disadvantages of Java SAX Parser −

- Random access to an XML document is not possible.
- Creating XML documents is not possible.
- If you want to keep track of data that the parser has seen or change the order of items, you must write the code and store the data on your own.

ContentHandler Interface

ContentHandler interface is the main interface in **org.xml.sax** package. Most of the application programs implement this interface to perform basic parsing events. These events include start and end of a document, start and end of the elements and character data. We must implement and register a Handler to perform any task in the XML document.

There are built-in classes, namely, **DefaultHandler**, DefaultHandler2, ValidatorHandler that implement ContentHandler interface. We can use these classes to implement our user defined Handlers.

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. Following are the methods of ContentHandler interface −

Method	**Description**
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

The **Attributes** interface is in the package **org.xml.sax**. This interface is for the list of XML attributes specified in an Element. Following are the most commonly used methods of Attributes interface −

Method	**Description**
int getLength()	Returns number of attributes.
int getIndex(String qName)	Returns the index of the attribute in the list, -1 if not present.
String getQName(int index)	Returns the attribute name by index, null if the index is out of bounds.
String getType(int index)	Returns the type ("CDATA", "ID", "IDREF", etc.) of attribute by index.
String getValue(int index)	Returns the value of attribute by index, null if index is out of bounds.
String getValue(String qName)	Returns the value of attribute by name, null if name is not present.