Java StAX Parser - Overview

StAX is a Java-based API to parse XML document in a similar way as SAX parser does. But unlike SAX parser, the StAX parser API is a simple iterator based API that gives parsing control to the client program. It reads the XML document in a forward-only fashion and stores the events in an iterator.

The client can ask for the events that he wants to access based on the event type of each event.

Differences Between SAX and StAX Parsers

Here are some notable differences between SAX and StAX parsers –

SAX Parser	StAX Parser
Push based stream parser	Pull based stream parser
XML documents cannot be created	XML documents cannot be created
Gives less parsing control to client program	Gives more parsing control to client program
Not Iterator based API	Iterator based API
Handler class must be implemented	No need to implement any Handler class
Provides schema validation	Doesn't provide schema validation

Environment Setup

In order to use StAX parser, you should have **stax.jar** in your application's classpath.

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

Features of StAX

Following are the features of StAX API –

- 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 StAX Parser?

You should use a StAX 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).

Disadvantages of StAX

Following are the disadvantages of StAX parser −

- We have no random access to an XML document, since it is processed in a forward-only manner.
- XML schema validation is not supported.
- If you need to keep track of data that the parser has seen or where the parser has changed the order of items, then you must write the code and store the data on your own.

XMLEvent Interface

This interface provides the basic Event representation of all components of an XML document. The event type differentiates each event and the information is retrieved accordingly. Some of the most commonly used methods of this interface are as follows −

Method	**Description**
StartElement asStartElement()	Retrieves StartElement object from this event.

EndElement asEndElement()	Retrieves EndElement object from this event.
Characters asCharacters()	Retrieves characters such as CDATA, whitespace, etc. from this event
int getEventType()	Returns the integer code for this event.

XMLEventReader Interface

This interface provides iterator of events which can be used to iterate over events as they occur while parsing an XML document. Some of the most commonly used methods are listed below −

Method	**Description**
boolean hasNext()	Returns true if there are more events in the EventReader, else retruns false.
XMLEvent nextEvent()	Returns the next XMLEvent in the EventReader.
XMLEvent peek()	Returns the next XMLEvent without reading it from the stream.

XMLEventWriter Interface

This interface writes XML documents by adding events. Some of the most commonly used methods are listed below −

Method	**Description**
void add(Event event)	Adds the event containing elements to XML.
void flush()	Writes any cached events to the stream.
void close()	Closes all resources related to the stream.

XMLStreamReader Interface

This interface provides efficient way of reading XML events in a forward, read-only manner. Some of its methods are listed below −

Method	**Description**

int next()	Used to retrieve next event.
boolean hasNext()	Used to check further events exists or not.
String getText()	Used to get text of the current parsing event.
String getLocalName()	Used to get local name of the current event.

XMLStreamWriter Interface

This Interface provides methods to write XML documents. Some of the most commonly used methods are listed below −

Method	Description
void writeStartDocument()	Adds XML declaration statement to the output stream.
void writeStartElement(String localName)	Adds a start element with given name.
void writeEndElement(String localName)	Adds an end element with given name.
void writeAttribute(String localName, String value)	Writes attribute with the specified name and value to an element.