APACHE XERCES SAX PARSER - OVERVIEW

SAX *theSimpleAPIforXML* 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***Stringuri, StringlocalName, StringqName, Attributesatts* - Called at the beginning of an element.
- **void endElement***Stringuri, StringlocalName, StringqName* - Called at the end of an element.
- **void characters***char[]ch, intstart, intlength* - Called when character data is encountered.
- **void ignorableWhitespace***char[]ch, intstart, intlength* - Called when a DTD is present and ignorable whitespace is encountered.

- **void processingInstruction***Stringtarget, Stringdata* - Called when a processing instruction is recognized.
- **void setDocumentLocator***Locatorlocator***)** - Provides a Locator that can be used to identify positions in the document.
- **void skippedEntity***Stringname* - Called when an unresolved entity is encountered.
- **void startPrefixMapping***Stringprefix, Stringuri* - Called when a new namespace mapping is defined.
- **void endPrefixMapping***Stringprefix* - 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***intindex*
- **String getValue***intindex*
- **String getValue***Stringqname*