Java DOM Parser - Overview

Java DOM parser is an API (Application Programming Interface) that has classes, interfaces and methods to parse XML documents by creating DOM tree structure.

The Document Object Model (DOM) is an official recommendation of the World Wide Web Consortium (W3C). It defines an interface that enables programs to access and update the style, structure, and contents of XML documents.

When to use Java DOM Parser?

You should use a DOM parser when −

- You need to know a lot about the structure of a document.
- You need to move parts of an XML document around (you might want to sort certain elements, for example).
- You need to use the information in an XML document more than once.

What is the result of Parsing?

When you parse an XML document with a DOM parser, you will get a **tree structure** that contains all of the elements of your document. The DOM provides a variety of functions you can use to examine the contents and structure of the document.

Advantages

Following are some advantages of Java DOM parser −

- DOM is a very simple API to use.
- Easy to access and modify any part of the document since entire document is loaded into memory.
- Java code written for one DOM-compliant parser should run on any other DOM-compliant parser without having to do any modifications.

Disadvantages

Following are some of the disadvantages of Java DOM parser −

- Consumes more memory
- Not suitable for large documents.
- Not applicable for small devices like PDAs and cellular phones.

DOM Interfaces

Here are the most commonly used DOM interfaces −

Interface	Description
Document	Represents the XML or HTML document.
Node	Primary data type of the Document Object Model (DOM)
NodeList	An ordered collection of nodes.
Element	Represents an element in XML document.
Attr	Represents an attribute of an Element object.
Text	Represents text content of an element or attribute, inherited from 'CharacterData' interface.

DOM Methods

Here are the most commonly used DOM methods −

Method	Description
Document.getDocumentElement()	This method returns the root element of the document.
Document.getElementsByTagName()	This method returns a NodeList of all elements within the given tag name.
Node.getFirstChild()	This method returns the first child of a given Node.
Node.getLastChild()	This method returns the last child of a given Node.
Node.getNextSibling()	This method returns the next sibling of the given node.

Node.getPreviousSibling()	This method returns the root element of the document.
Node.getAttribute(attrName)	This method returns the attribute with the requested name for a given node.
Node.getNodeName()	This method returns the name of the node.
Node.getTextContent()	This method returns the text content of the current node and its descendants.
NodeList.getLength()	This method returns the number of nodes in the NodeList.
NodeList.item(int index)	This method returns the indexth node from the NodeList.