SAX can parse XML documents but cannot create them. In contrast, DOM can parse and create XML documents. This chapter introduces you to DOM.

1. **What is DOM?**

Document Object Model (DOM) is a Java API for parsing an XML document into an in-memory tree of nodes, and for creating an XML document from a node tree. After a DOM parser creates a tree, an application uses the DOM API to navigate over and extract infoset items from the tree’s nodes.

DOM has two big advantages over SAX:

-DOM permits random access to a document’s infoset items, whereas SAX only permits serial access.

-DOM also lets you create XML documents, whereas you can only parse documents with SAX.

1. **A Tree of Nodes**

DOM views an XML document as a tree that is composed of several kinds of nodes.

DOM views an XML document as a tree that is composed of several kinds of nodes.

DOM classifies nodes into 12 types: Attribute node, CDATA section node, Comment node, Document node, Document fragment node, Document type node, Element node,

Entity node, Entity reference node, Notation node, Processing instruction node, Text node

1. **Exploring the DOM API**

**3.1. Obtaining a DOM Parser/Document Builder**

A DOM parser is also known as a document builder because of its dual role in parsing and creating XML documents. You obtain a DOM parser/document builder by first instantiating ocumentBuilderFactory, by calling one of its newInstance() class methods.

After obtaining a DocumentBuilderFactory instance, you can call various configuration methods to configure the factory.

After the factory has been configured, call its DocumentBuilder newDocumentBuilder() method to return a document builder that supports the configuration

**3.2. Parsing and Creating XML Documents**

DocumentBuilder provides several overloaded parse() methods for parsing an XML document into a node tree

Node declares 12 constants that represent the various kinds of nodes; ATTRIBUTE\_NODE and ELEMENT\_NODE are examples

Node declares several methods for navigating the node tree. Three of its navigation methods: boolean hasChildNodes(), Node getFirstChild(), Node getLastChild()

Node declares four methods for modifying the tree by inserting, removing, replacing, and appending child nodes: Node insertBefore (Node newChild, Node refChild), Node removeChild (Node oldChild), Node replaceChild (Node newChild, Node oldChild), Node appendChild (Node newChild)

Node declares several utility methods.

Document declares three methods for locating one or more elements: Element getElementById(String elementId), NodeList getElementsByTagName(String tagname), NodeList getElementsByTagNameNS(String namespaceURI, String localName)

The returned element node and each element node in the list implement the Element interface. This interface declares methods to return nodelists of descendent elements in the tree, attributes associated with the element, and more.