JAVA XML AND JSON

Chapter 3: Parsing and Creating XML Documents with 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.

1. A Tree of Nodes

DOM views an XML document as a tree that is composed of several kinds of nodes. This tree has a singleroot nodeand all nodes, except for the root, have aparent node.

Each node has a node name, which is the complete name for nodes that have names (such as an element’s or an attribute’s prefixed name), and #node-type for unnamed nodes.

DOM classifiesnodes into 12 types, of which seven types can be considered part of a DOM tree:

* 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

Java implements DOM through the javax.xml.parsers package’s abstract DocumentBuilder and DocumentBuilderFactory classes and the nonabstract FactoryConfigurationError and ParserConfigurationException classes.

1. Obtaining a DOMParser/Document Builder

* A DOM parser (document builder) parses and creates XML documents. You obtain a DOM parser/document builder by first instantiating DocumentBuilderFactory, by calling one of its newInstance() class methods (Ex: DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();).
* Behind the scenes, newInstance() follows an ordered lookup procedure to identify the DocumentBuilderFactory implementation class to load.
* If an implementation class isn’t available or cannot be instantiated, newInstance() throws an instance of the FactoryConfigurationError class. Otherwise, it instantiates the class and returns its instance.
* 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 (DocumentBuilder db = dbf.newDocumentBuilder();)
* If a document builder cannot be returned, this method throws a ParserConfigurationException instance.

1. Parsing and Creating XML Documents

* DocumentBuilder provides several overloaded parse() methods for parsing an XML document into a node tree. These methods differ in how they obtain the document. DocumentBuilder also declares the abstract Document newDocument() method for creating a document tree.
* Node declares 12 constants that represent the various kinds of nodes. To identify the kind of node represented by a given Node object, call Node’s short getNodeType() method and compare the returned value to one of these constants.
* Node declares several methods for getting and setting common node properties (Ex: String getNodeName(), String getLocalName(),...).
* Node declares several methods for navigating the node tree. Three of them are:
* boolean hasChildNodes() returns true when a node has child nodes.
* Node getFirstChild() returns the node’s first child.
* Node getLastChild() returns the node’s last child.
* The NodeList getChildNodes() method returns an org.w3c.dom.NodeList instance whose int getLength() methodreturns the number of nodes in the list, and whose Node item(int index) method returns the node at the indexth position in the list.
* Node declares four methods for modifying the tree by inserting, removing, replacing, and appending child nodes :
* Node insertBefore (Node newChild, Node refChild) inserts newChild before the existing node specified by refChild and returns newChild.
* Node removeChild (Node oldChild) removes the child node identified by oldChild from the tree and returns oldChild.
* Node replaceChild (Node newChild, Node oldChild) replaces oldChild with newChild and returns oldChild.
* Node appendChild (Node newChild) adds newChild to the end of the current node’s child nodes and returns newChild.
* As well as inheriting Node’s constants and methods, Document declares its own methods. Document declares three methods for locating one or more elements:
* Element getElementById(String elementId) returns the element that has an id attribute matching the value specified by elementId.
* NodeList getElementsByTagName(String tagname) returns a nodelist of a document’s elements matching the specified tagName.
* NodeList getElementsByTagNameNS(String namespaceURI,String localName) is equivalent to the second method except in adding to the nodelist only those elements matching localName and namespaceURI values.
* The returned element node and each element node in the list implement the Element interface.