Chapter 6: Java-Libraries for Processing XML: DOM and SAX

References:

- The Java Tutorials: Java API for XML Processing [https://docs.oracle.com/javase/tutorial/jaxp/index.html]
- Java Platform Documentation: Package `javax.xml.parsers` [https://docs.oracle.com/javase/7/docs/api/javax/xml/parsers/package-summary.html]
- Java Platform Documentation: Package `org.w3c.dom` [http://docs.oracle.com/javase/7/docs/api/org/w3c/dom/package-summary.html]
- W3C Homepage for the Document Object Model (DOM) [https://www.w3.org/DOM/]
- W3C: Document Object Model (DOM) Level 3 Core Specification [https://www.w3.org/TR/DOM-Level-3-Core/]
- Official Website for SAX: [http://www.saxproject.org/]
- Java Platform Documentation: Package `org.xml.sax` [http://docs.oracle.com/javase/7/docs/api/org/xml/sax/package-summary.html]
- [https://www.tutorialspoint.com/java_xml/java_dom_parser.htm]
- [https://www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/]

Objectives

After completing this chapter, you should be able to:

- write Java programs that process XML data using the DOM and SAX libraries.

  Since this is only a very short chapter, it is ok if you have only a general impression how to do this and are not yet completely fluent writing such programs.

Overview

Introduction (1)

- Of course, the main focus of this course is
  - ◇ defining the structure of XML data with XML schema, and
  - ◇ querying XML data with XPath and XQuery.

    XPath 2.0 is a quite powerful subset of XQuery 1.0.
- Sometimes, using XML is only part of a larger application program. Then one needs programmatic access to the structured data in an XML file.

Introduction (2)

- There are a number of different interfaces that can be used to access XML data from Java:
  - ◇ DOM (Document Object Model)

    This is a programming-language independent standard from the W3C. It has many parts that are mainly used for manipulating HTML in JavaScript, but the DOM Core is applicable to XML as well. It is similar to the XPath/XQuery Data Model from Chapter 5, but it has also methods for modifying the XML tree.

  - ◇ SAX (Simple API for Java)

    If the data is too big to be kept completely in main memory, this event-based interface allows to travers the tree without actually constructing it.

Introduction (3)

- There are more APIs for using Java Parsers besides DOM and SAX:
  - ◇ StAX: Streaming API for XML

    This is similar to SAX, but instead of registring callback functions as in SAX for parser events, one asks an iterator for the next event. Thus, the client program is in control ("pull"), whereas in SAX, the parser is in control ("push"). StAX also allows to write XML documents, whereas SAX is read only.
  - ◇ JDOM [http://www.jdom.org/]

    This claims to be a simpler object-oriented interface to XML in Java than DOM. One reason is that it was designed specifically for Java, whereas DOM is language-independent.
  - ◇ XOM [http://www.xom.nu/]

Overview

Overview

SAX Example (1)

```
(1)   import org.xml.sax.XMLReader;
(2)   import org.xml.sax.Attributes;
(3)   import org.xml.sax.InputSource;
(4)   import org.xml.sax.helpers.XMLReaderFactory;
(5)   import org.xml.sax.helpers.DefaultHandler;
(6)   import org.xml.sax.SAXException;
(7)   import org.xml.sax.SAXParseException;
(8)   import java.io.FileInputStream;
(9)   import java.io.InputStreamReader;
(10)  import java.io.IOException;
(11)
(12)  public class SaxExample
(13)      extends DefaultHandler {
```

SAX Example (2)

```
(14)   public static void main(String args[]) {
(15)      // Create XML Reader:
(16)      XMLReader xr = null;
(17)      try {
(18)         xr = XMLReaderFactory.
(19)                createXMLReader();
(20)         SaxExample handler = new SaxExample();
(21)         xr.setContentHandler(handler);
(22)         xr.setErrorHandler(handler);
(23)      } catch(Exception e) {
(24)         System.err.println("Exception " +
(25)                "creating XMLReader: " + e);
(26)         System.exit(1);
(27)      }
```

SAX Example (3)

```
(28)
(29)        // Open Input File:
(30)        String file = "test.xml";
(31)        String code = "ISO-8859-1";
(32)        InputStreamReader in = null;
(33)        try {
(34)           FileInputStream is =
(35)                 new FileInputStream(file);
(36)           in = new InputStreamReader(is, code);
(37)        } catch(IOException e) {
(38)           System.err.println("Error opening " +
(39)                 "'" + file + "': " + e);
(40)           System.exit(2);
(41)        }
```

SAX Example (4)

```
(42)
(43)        // Parse the XML input file:
(44)        try {
(45)            xr.parse(new InputSource(in));
(46)        } catch(Exception e) {
(47)            System.err.println("Error parsing " +
(48)                "'" + file + "': " + e);
(49)            System.exit(3);
(50)        }
(51)
(52)    } // End of main
(53)
```

SAX Example (5)

```
(54)   //=====================================
(55)   // Event Handlers:
(56)   //=====================================
(57)
(58)   public void startDocument() {
(59)      System.out.println("Start document");
(60)   }
(61)
(62)   public void endDocument() {
(63)      System.out.println("End document");
(64)   }
(65)
```

SAX Example (6)

```
(66)   public void startElement(String uri,
(67)                            String name, String qName,
(68)                            Attributes attrs) {
(69)
(70)      String elem;
(71)      if ("".equals(uri))
(72)         elem = qName;
(73)      else
(74)         elem = "{" + uri + "}" + name;
(75)      System.out.println("Element: " + elem);
(76)
```

SAX Example (7)

```
(77)        int attrCount = attrs.getLength();
(78)        if(attrCount>0) {
(79)           System.out.println("Attributes:");
(80)           for(int i = 0 ; i < attrCount; i++) {
(81)              System.out.println(" Name : " +
(82)                 attrs.getQName(i));
(83)              System.out.println(" Type : " +
(84)                 attrs.getType(i));
(85)              System.out.println(" Value: " +
(86)                 attrs.getValue(i));
(87)              System.out.println();
(88)           }
(89)        }
(90)     }
```

SAX Example (8)

```
(91)
(92)    public void endElement(String uri,
(93)                           String name, String qName) {
(94)        if(name == null || name.equals(""))
(95)            name = qName;
(96)        System.out.println("End Elem: " + name);
(97)    }
(98)
```

SAX Example (9)

```
 (99)   public void characters(char ch[],
(100)                  int start, int length) {
(101)      System.out.print("Text: \"");
(102)      for (int i = start; i < start + length;
(103)                              i++) {
(104)         System.out.print(ch[i]);
(105)      }
(106)      System.out.println("\"");
(107)   }
```

SAX Example (10)

```
(108)   public void warning(SAXParseException e) {
(109)      System.err.println("Warning at line " +
(110)                 e.getLineNumber());
(111)      System.err.println(e.getMessage());
(112)   }
(113)
(114)   public void fatalError(SAXParseException e)
(115)                    throws SAXException {
(116)      System.err.println("Fatal error, line " +
(117)                 e.getLineNumber());
(118)      System.err.println(e.getMessage());
(119)      throw e;
(120)   }
```