

X**M****L**

Data Management

dott. Chris Mair
2009/2010 @unibz

5 – JAVA APIs for XML

last modified 2009-11-04

Parsing strategies

Event-based parsers

- scanner produces a stream of XML tokens
- events can be triggered by tokens
- events happen in lexical order - random access is *not* possible
- the standard API for event-based XML processing is **SAX** (Simple **A**PI for **X**ML) – see <http://www.saxproject.org>

Tree-based parsers

- parser produces a tree data structure
- data structure completely held in-memory
- random access *is possible*
- the standard interface to access such an in-memory representation is W3C's **DOM** (**D**ocument **O**bject **M**odel) – see <http://www.w3.org/DOM>

SAX example

```

1  /*
2     SAXTest.java
3
4     (loosly based on the SAXCounter example in
5     "Learning XML, 2nd Edition")
6  */
7
8  import org.xml.sax.helpers.DefaultHandler;
9  import org.xml.sax.Attributes;
10 import org.xml.sax.XMLReader;
11 import org.xml.sax.helpers.XMLReaderFactory;
12 import org.xml.sax.InputSource;
13 import java.io.FileReader;
14
15 public class SAXTest extends DefaultHandler {
16
17     private int elements;
18
19     public static void main (String args[]) throws Exception {
20
21         // get a Reader object
22         XMLReader xr = XMLReaderFactory.createXMLReader();
23
24         // set an instance of ourselves as the handler (i.e. the object
25         // that is called back whenever the parser creates an event)
26         xr.setContentHandler(new SAXTest());
27
28         // parse the file given as command line argument
29         xr.parse(new InputSource(
30             new FileReader("/Users/chris/prj/unibz/xml/t/recipe.xml")));
31     }
32 }
33
34 public SAXTest () {
35     super();
36 }
37
38 // handle a start-of-document event
39 public void startDocument ()
40 {
41     System.out.println("Starting to parse...");
42     elements = 0;
43 }
44
45 // handle an end-of-document event
46 public void endDocument ()
47 {
48     System.out.println("All done!");
49     System.out.println("There were " + elements + " elements.");
50 }
51
52 // handle a start-of-element event
53 public void startElement (String uri, String name,
54     String qName, Attributes atts) {
55     System.out.println("START of element " + qName);
56     if (!"".equals(uri)) {
57         System.out.println("    namespace: " + uri);
58     }
59     System.out.println("    number of attributes: " + atts.getLength());
60 }
61

```

```
62 // handle an end-of-element event
63 public void endElement (String uri, String name, String qName)
64 {
65     elements++;
66     System.out.println("END of element " + qName);
67 }
68
69 // handle a characters event
70 public void characters (char ch[], int start, int length)
71 {
72     int i;
73     System.out.print("CDATA: \"");
74     for (i = start; i < start + length; i++) {
75         if (ch[i] == '\n') {
76             System.out.print("\n");
77         } else {
78             System.out.print(ch[i]);
79         }
80     }
81     System.out.println("\"");
82 }
83
84 }
85
86
```

DOM example

```

1  /*
2     DOMTest.java
3
4  */
5
6  import javax.xml.parsers.DocumentBuilder;
7  import javax.xml.parsers.DocumentBuilderFactory;
8  import java.io.File;
9  import org.w3c.dom.Document;
10 import org.w3c.dom.Node;
11 import org.w3c.dom.NodeList;
12
13
14 public class DOMTest {
15
16     static Document document;
17
18     public static void main(String[] argv) throws Exception {
19
20         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
21
22         // factory.setValidating(true);
23         // factory.setNamespaceAware(true);
24
25         DocumentBuilder builder = factory.newDocumentBuilder();
26         document = builder.parse(new File("/Users/chris/prj/unibz/xml/t/recipe.xml"));
27         //System.out.println(document.getDocumentElement().getNodeName());
28
29         // let's call our traversal code
30         traverse(document.getDocumentElement(), 0);
31
32     }
33
34     private static final int OPEN=1;
35     private static final int CLOSE=2;
36
37
38     private static void traverse(Node n, int level) {
39         int i;
40
41         // find out what type of node this is
42
43         short t = n.getNodeType();
44
45         if (t == Document.ELEMENT_NODE) {
46             print_element(level, n, OPEN);
47         } else if (t == Document.TEXT_NODE) {
48             if (!is_empty(n)) {
49                 print_text(level, n);
50             }
51         }
52
53         NodeList nl = n.getChildNodes();
54         if (nl.getLength() == 0) {
55             return;
56         }
57
58         for (i = 0; i < nl.getLength(); i++) {
59             traverse(nl.item(i), level + 1);
60         }
61
62         if (t == Document.ELEMENT_NODE) {
63             print_element(level, n, CLOSE);
64         }
65
66     }
67
68     static void print_element(int level, Node n, int mode) {
69         String slash = "";
70         if (mode == CLOSE) {
71             slash = "/";
72         }

```

```
73     System.out.println(get_indent(level) + "<" + slash + n.getNodeName() + ">");
74 }
75 static void print_text(int level, Node n) {
76     System.out.println(get_indent(level) + n.getNodeValue());
77 }
78
79
80 static String get_indent(int level) {
81     int i;
82     StringBuffer buf = new StringBuffer();
83     for (i = 0; i < level; i++) {
84         buf.append("    ");
85     }
86     return buf.toString();
87 }
88
89 static boolean is_empty(Node n) {
90
91     String val = n.getNodeValue();
92     val = val.replaceAll("\\s+", "");
93     val = val.replaceAll("\u00a0+", "");
94     if (val.equals("")) {
95         return true;
96     }
97     return false;
98
99 }
100
101 }
102
103
104
```