

SAX XML Parsing

SAX parsing is cheaper than DOM parsing -- it tells you about each element as it is found in a single pass of the XML. We must maintain any state during the parse ourselves.

```
// XMLDotReader.java
// Uses the SAX interface
import java.io.*;
import java.util.*;

import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;

/**
 * This is a simple class that can read state out of an XML file
 * using a SAX state-machine parser.
 *
 * http://java.sun.com/xml/
 *
 * In this case, we support data like this, where the flip
 * node switches x,y...
 *
 * <?xml version="1.0" encoding="UTF-8"?>
 *
 * <dots>
 *   <dot x="81" y="67" />
 *   <dot x="175" y="122" />
 *   <flip>
 *     <dot x="175" y="122" />
 *     <dot x="209" y="71" />
 *   </flip>
 *   <dot x="209" y="71" />
 * </dots>
 */
public class XMLDotReader extends DefaultHandler
{
    public static void main (String argv [])
    {
        if (argv.length != 1) {
            System.err.println ("Usage: cmd filename");
            System.exit (1);
        }

        try {
            XMLDotReader xr = new XMLDotReader();

```

```

        InputStream in = new BufferedInputStream( new FileInputStream(new
File(argv[0])));
        xr.read(in);
    } catch (Throwable t) {
        t.printStackTrace ();
    }
}

/**
Read the XML in the given file
*/
public void read(InputStream stream) {
    try {
        SAXParserFactory factory = SAXParserFactory.newInstance();
        SAXParser saxParser = factory.newSAXParser();
        clear();
        saxParser.parse(stream, this);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}

public XMLDotReader() {
    clear();
}

// State we keep track of -- like a state machine,
// where startElement() etc. keep getting called
private int x;
private int y;
private boolean flip;

public void clear() {
    x = -1;
    y = -1;
    flip = false;
}

//=====
// SAX DocumentHandler methods
//=====

public void startDocument ()
throws SAXException
{
    //System.out.println("startDocument");
}

public void endDocument ()
throws SAXException
{
    //System.out.println("startDocument");
}

```

```

}

/**
 Called for each node
 -look at qName and atts to see the node state
 -process that node if appropriate
 -or, update our state to affect future calls to startElem()
 or characters()
 */
public void startElement (String namespaceURI, String localName,
                        String qName, Attributes atts)
throws SAXException
{
    //System.out.println("start element:" + qName);
    if (qName.equals("dot")) {
        x = Integer.parseInt(atts.getValue("x"));
        y = Integer.parseInt(atts.getValue("y"));

        if (flip) {
            int temp = x; x = y; y = temp;
        }

        // do something with our x,y state (could wait for endElement)
        System.out.println(x + ", " + y);
    }
    else if (qName.equals("flip")) {
        flip = true;
    }
}

// Called at the end of each element
public void endElement(java.lang.String uri,
                      java.lang.String localName,
                      java.lang.String qName)
throws SAXException
{
    //System.out.println("end element:" + localName);

    if (qName.equals("flip")) {
        flip = false;
    }
}

// Called for characters between nodes.
// May be called multiple times for the text within tag --
// once per line for example.
public void characters (char buf [], int offset, int len)
throws SAXException
{
    //String s = new String(buf, offset, len);
    //s = s.trim();
    //if (!s.equals("")) {
        //System.out.println("characters:" + s);
    //}
}
}

```