CS 104: Web Technology - II Assignment #5

Anup Adhikari
anup.adhikari@gandakiuniversity.edu.np

Gandaki University January 25, 2023

1 Introduction to SAX (Simple API for XML)

Building and traversing a large DOM tree can waste time and memory.

For this reason a second standard API exists for event-based XML parsers.

A SAX parser traverses an XML document without building a tree.

Every time a SAX parser enters an element node, it calls the startElement method of a given event handler. When it exits the node the handler's endElement method is called. When a text element is encountered, the handler's characters method is called.

Java provides a DefaultHandler class that implements these and other methods as empty operations.

Programmers simply create extensions of the DefaultHandler class and override those methods of interest.

Here's the basic design:

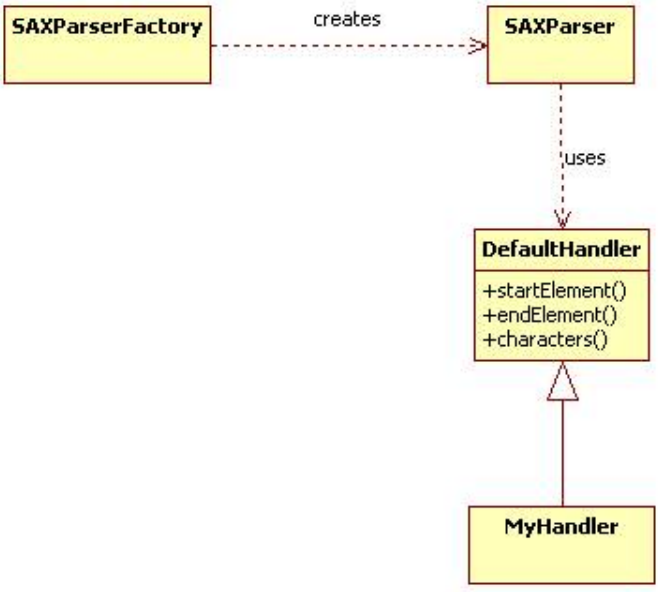

Figure 1: SAX

1.1 When SAX is better than DOM

- It parses the XML file as a stream rather than allocating RAM for the complete file.
- Since, it uses less memory and is faster than the DOM Parser because the complete file is not stored in memory.
- Therefore, it is considered to be useful in parsing large XML files.

1.2 Drawbacks of SAX over DOM

- There are no update methods in the SAX Parser. Since the complete file isn't kept in memory, it is possible to access items only in a sequential manner and the elements cannot be accessed randomly.

1.3 Required Imports

saxParser.java

```
import javax.xml.parser.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.io.*;
```

2 Objectives

The Objectives of lab are:

- To understand the basic implementation of SAX.

3 Examples

The static visit method in SAXUtils.java demonstrates how an XML file is parsed using SAX Parser.

Each time an element node is entered by the parser, the startElement method of the handler is called. When the element node is exited, the endElement node is called.

When a text node is encountered, the handler's characters method is called.

These methods are no-ops in the default handler, but they may be selectively overridden in subclasses.

SAXUtils.java

```
import org.xml.sax.helpers.*;
import javax.xml.parsers.*;
import org.xml.sax.*;
import java.io.*;

public class SAXUtils {

    public static void visit(String xmlFile, DefaultHandler handler) {
        SAXParserFactory factory = SAXParserFactory.newInstance();
        try {
            SAXParser saxParser = factory.newSAXParser();
            saxParser.parse( new File(xmlFile), handler );
        }
        catch (Exception e) {
            System.out.println(e);
        }
    }

}
```

3.1 A Pretty Print Handler

The file PrettyPrintHandler.java shows how the default handler is typically extended.

PrettyPrintHandler.java

```
import org.xml.sax.*;
import org.xml.sax.helpers.*;

public class PrettyPrintHandler extends DefaultHandler {

    private String prefix = ".............................";
    private int depthCounter = 0;

    public void startElement(
        String namespaceURI,
        String localName, // simple name
        String qName, // qualified name
        Attributes attrs) throws SAXException {
            depthCounter++;
            String start = prefix.substring(0, depthCounter);
            System.out.println(start + qName);

            depthCounter++;
            start = prefix.substring(0, depthCounter);
            for(int i = 0; i < attrs.getLength(); i++) {
            System.out.println(start + attrs.getQName(i)
            + " = " + attrs.getValue(i));
        }
        depthCounter--;
    }

    public void endElement(String uri, String localName, String qName) {
        depthCounter--;
    }

    public void characters(char buf[], int offset, int len) throws SAXException {
        String text = new String(buf, offset, len);
        depthCounter++;
            String start = prefix.substring(0, depthCounter);
            System.out.println(start + text);
            depthCounter--;
    }

    }
```

Here is a simple test driver.

TestSAX.java

```
public class TestSAX {
    public static void main(String[] args) {
        SAXUtils.visit("org1.xml", new PrettyPrintHandler());
    }
}
```

org1.xml

```
<?xml version="1.0"?>
<org>
    <member id="p1" gender="male">
        <lastName>Simpson</lastName>
        <firstName>Homer</firstName>
        <dob>1952-07-04</dob>
        <spouse id="p2" />
        <child id="p3" />
        <child id="p4" />
        <child id="p5" />
    </member>
    <dependant id="p2" gender="female">
        <lastName>Simpson</lastName>
        <firstName>Marge</firstName>
        <dob>1955-10-20</dob>
        <sponsor id="p1" />
    </dependant>
    <dependant id="p3" gender="female">
        <lastName>Simpson</lastName>
        <firstName>Lisa</firstName>
        <dob>1985-06-22</dob>
        <sponsor id="p1" />
    </dependant>
    <dependant id="p4" gender="female">
        <lastName>Simpson</lastName>
        <firstName>Maggie</firstName>
        <dob>1988-11-11</dob>
        <sponsor id="p1" />
    </dependant>
    <dependant id="p5" gender="male">
        <lastName>Simpson</lastName>
        <firstName>Bart</firstName>
        <dob>1983-01-01</dob>
        <sponsor id="p1" />
    </dependant><!-- etc. -->
</org>
```

3.2 Finding Sponsors

TestSAX.java

```
private Map<String, String> members = new Hashtable<String, String>();
   private Map<String, String> dependants = new Hashtable<String, String>();

    // merge tables
   public Map<String, String> getSponsors() {
      Map<String, String> result = new Hashtable<String, String>();
      Set<String> dependantNames = dependants.keySet();
      for(String dn: dependantNames) {
         String sponsorName = members.get(dependants.get(dn));
         result.put(dn, sponsorName);
      }
      return result;
   }
```

3.3 Skipping Nodes with SAX

Main Program

Demo.java

```
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.XMLReader;

public class Demo {

    public static void main(String[] args) throws Exception {
        SAXParserFactory spf = SAXParserFactory.newInstance();
        SAXParser sp = spf.newSAXParser();
        XMLReader xr = sp.getXMLReader();
        xr.setContentHandler(new MyContentHandler(xr));
        xr.parse("input.xml");
    }
}
```

MyContentHandler

MyContentHandler.java

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;

public class MyContentHandler extends DefaultHandler {

    private XMLReader xmlReader;

    public MyContentHandler(XMLReader xmlReader) {
        this.xmlReader = xmlReader;
    }

    public void startElement(String uri, String localName, String qName,
            Attributes atts) throws SAXException {
        if ("sodium".equals(qName)) {
            xmlReader.setContentHandler(new IgnoringContentHandler(xmlReader,
                    this));
        } else {
            System.out.println("START " + qName);
        }
    }

    public void endElement(String uri, String localName, String qName)
            throws SAXException {
        System.out.println("END " + qName);
    }

    public void characters(char[] ch, int start, int length)
            throws SAXException {
        System.out.println(new String(ch, start, length));
    }

}
```

When the IgnoringContentHandler is done swallowing events it passes control back to your main ContentHandler.

Ignoring Content Handler

IgnoringContentHandler.java

```
import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;

public class IgnoringContentHandler extends DefaultHandler {

    private int depth = 1;
    private XMLReader xmlReader;
    private ContentHandler contentHandler;

    public IgnoringContentHandler(XMLReader xmlReader, ContentHandler contentHandler) {
        this.contentHandler = contentHandler;
        this.xmlReader = xmlReader;
    }

    public void startElement(String uri, String localName, String qName,
            Attributes atts) throws SAXException {
        depth++;
    }

    public void endElement(String uri, String localName, String qName)
            throws SAXException {
        depth--;
        if(0 == depth) {
            xmlReader.setContentHandler(contentHandler);
        }
    }

}
```