# CMPSC380 Final Project: Advanced Topic in Data Management
# XML Applications with DOM and SAX

Andreas Bach Landgrebe[1]

[1]Allegheny College, Department of Computer Science

Thursday, December 12th, 2014

# 1 Introduction

Comparing and Contrasting DOM and SAX: Picking the best parsing method

What is DOM?

DOM is a document object model. This is a cross platform and language-independent convention for representing and interacting with objects in XML documents. It is also able to interact with objects in HTML and XHTML objects. For the purposes of this project, I will be using DOM to be able to interact with XML documents. This parser is a tree-based API.

What is SAX?

SAX is a Simple API for XML. This is an event sequential access parser API for XML documents. This provides a mechanism for reading data from an XML document that is an alternative to that provided by the Document Object Model (DOM). The SAX parsers operates on each piece of the XML document sequentially while the DOM operates on the document as a whole. This parser is an event-based API.

# 2 Comparing and Contrasting

DOM (Strengths, Weaknesses, Features)

### Features

1. It a hierarchy-based parser that creates an object model of the entire XML document.

2. XML will be broken down into three main pieces: Elements (sometimes called tags), Attributes, and the data that the elements and attributes describe.

### Strengths

- DOM is able to support random-access manipulation. One will be able to see the tokens more than once in random order.

- DOM XML parsing is able to retain a complete model which is able to prevent reduced memory overhead.
- The DOM approach discards more debugging output than an approach like SAX (Simple API for XML).
- The DOM parser is not as challenging to implement compared to the SAX parser
- The DOM parser is able to provide a document representation to be able to manipulate, serialize and traverse XML documents.

**Weaknesses**

- An approach like SAX presents a document as a serialized "event stream" which DOM does not do.
- The SAX-based XML parsing is able to scan and parse gigabytes worth of XML document without hitting resource limits since it does not try to create a DOM representation in memory.
- Generally, the SAX-based XML required fewer resources.
- If there is not available memory to be able parse such a large document, then the SAX parser will be the only approach to be able to parse your document.

SAX (Strengths, Weaknesses Features)
**Features**

- Parses the document on node by node basis
- Does not store the entire XML file in memory
- The SAX parses model uses top to bottom traversing

**Strengths**

- Consumes less memory from the DOM parsing method.
- It is easier to handle large XML structure using the SAX parsing versus the DOM parsing.
- You can parse a document in small contiguous chunks of input instead of parsing the entire document.

**Weaknesses**

- SAX XML parser are read-only so you cannot use this method to create a new XML document
- The SAX model does not preserve comments.
- This approach is more complex and tedious to implement compared to the DOM parsing method.
- If your application has to modify the document repeatedly, then the DOM parser is a better approach.

## 3 Implementation

### DOM vs SAX

1. The Easier Approach to Implement
   When completing this final project, after reading the source code several times, I began to notice a difference between the DOM and SAX based parsing approach. When the step came to implement the source code to be able to parse the document with these two approaches, it seemed to me that the DOM based approach was easier to implement versus the SAX based parsing approach for XML partly because call back implementation was new and because your program logic has to accommodate the structure of the XML file.

2. Complexity
   When looking at the complexity, I found some evident information to be able to explain why it is the case that the SAX parsing method is a more complex approach to implement. I found this to be the case since DOM will represent an XML document into a tree format. This will make it easier to read since the XML document has been generated into a tree format. The SAX parsing method is a little more complex. The SAX parsing XML method is a event-based so you will have to specify what event is going to trigger to start of the parsing. Since one has to specify this, there is more complexity with the approach of SAX.

## 4 Imperial Results

**Running books.xml (in seconds)**

| | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 |
|---|---|---|---|---|---|
| DOM | 0.208523 | 0.041428 | 0.042919 | 0.039658 | 0.045261 |
| SAX | 0.064641165 | 0.066524827 | 0.083815583 | 0.179422232 | 0.044658071 |

**Running ActorPreludeSample.xml (in seconds)**

| | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 |
|---|---|---|---|---|---|
| DOM | 3.543439 | 2.926811 | 4.833665 | 2.611887 | 3.087350 |
| SAX | 1.874940386 | 2.697318543 | 1.327047531 | 1.183420283 | 1.106874209 |

**Running EPAXMLDownload.xml (in seconds)**

| | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 |
|---|---|---|---|---|---|
| DOM | 41.001281 | 40.768404 | 38.8329267 | 39.084435 | 38.550187 |
| SAX | 2.298556144 | 2.292585293 | 2.358654313 | 2.317283911 | 2.325312500 |

**Running fibonacci.xml (in seconds)**

| | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 |
|---|---|---|---|---|---|
| DOM | 18.732033364 | 7.188593838 | 7.188286017 | 7.154550480 | 7.148989056 |
| SAX | 6.156964747 | 3.108887500 | 3.201726764 | 3.281493254 | 3.139322430 |

**Average of the two parsing methods**

1. Which one had the best performance?
   **Average Times of XML Files Being Run 5 Times (in seconds)**

| | Small | Medium | Large | Extra Large |
|---|---|---|---|---|
| DOM | 0.0755578 | 3.40081138 | 39.64744674 | 9.482490551 |
| SAX | 0.0878123756 | 1.6379201904 | 2.3184784316 | 3.777678939 |

**Small:** books.xml
**Medium:** ActorPreludeSample.xml
**Large:** EPAXMLDownload
**Extra Large:** fibonacci.xml

In the above figure, I decided to test the different parsing methods of DOM and SAX by using 4 different xml files. The xml file that is labelled Small in the table has the size of 4.4 kB (4430 bytes). The second xml file that is labelled Medium in the table has the size of 2.4 MB (1,294,565 bytes). The third xml file that is labelled Large in the table has the size of 237.5 MB (237,497,525 bytes). The last xml file that was being used to test the different parsing methods that is labelled Extra Large has the size of 1.0 GB (1,049,052,687 bytes).

2. Why is it the case of these results?

   According to the results that that I received from conducting my experiments, the SAX parsing method was the quicker parsing method. This is the case because of the fact that DOM parsing means that the whole XML file will be stored in memory. There may not be a big difference with the smaller xml files, but the xml files got larger and larger, I was able to see a big difference between how much time DOM and SAX took to parse and XML file. Loading an entire XML into memory to represent the tree can be time and memory consuming.

   Another trend I noticed is that it took a shorter amount of time to parse the extra large xml file called fibonacci.xml compared to the large xml file called EPAXMLDownload.xml. This is the case due to the complexity of the xml files. In the fibonacci, the fibonacci numbers are being calculated and this is the only piece of information that is being read by the parsing methods. In the EPAXMLDownload.xml file, there is a bit more complexity in this xml file. There are also more lines of code to parse through the EPAXMLDownload.xml file. In the fibonacci.xml file, there are 10,0003 lines of code to read in to be parsed. In the EPAXLDownload.xml file, there is 3,601,339 lines of code to be read in and to be parsed. However, despite how many lines of code are in each one of these xml files, the fiboancci.xml is large in size due to the size of the fibonacci numbers on each line. So the fibonacci.xml file may be larger in size due to the length of each line of code, but the EPAXMLDOwnload.xml file is longer in length considering how many lines of code which is the reason for having the EPAXMLDownload.xml file take a longer amount of time to parse.

   I also noticed that the first time that I ran the DOM parsing method, it took a sufficiently longer time to parse versus the rest of the times. This is the case because when looking at the DOM parsing method, there is an important feature to put in consideration. This feature is that the DOM parsing method loads the entire XML file into memory. This means that DOM is using a large amount of memory. Due to this, the memory has now seen a XML file that has been parsed and some information is still stored in memory to have the parsing take a shorter amount of time.

3. If SAX was quicker, why would you use DOM?
   When looking at the results, the performance of the SAX parsing method is better overall. If this is the case, the question to ask is when would use the DOM parsing method. Despite the speed was quicker with the SAX parsing method, there are benefits to using the DOM parsing method. One of these benefits to using DOM is altering the contents of the document. Using the DOM parsing method allows you to easily query any part of the document and freely manipulate all of the nodes in the tree. People also use DOM typically of smaller sizes XML structures. People generally use this parsing method to be able modify and query in different ways once it has been loaded. As a summary, one would use DOM to give you full flexibility in changing structure and contents, with the disadvantage of using more resources such as memory

## 5 Source Code

```
DOMEcho.java

// JAXP packages
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import org.w3c.dom.*;

import java.io.*;


public class DOMEcho {
    /** All output will use this encoding */
    static final String outputEncoding = "UTF-8";

    /** Output goes here */
    private PrintWriter out;

    /** Indent level */
    private int indent = 0;

    /*as a default, if the parameter -no is not called, then the
        debugging output
    will be displayed
    */
    static boolean noDebuggingOutput = false;

    /** Indentation will be in multiples of basicIndent */
    private final String basicIndent = " ";

    /** Constants used for JAXP 1.2 */
    static final String JAXP_SCHEMA_LANGUAGE =
    "http://java.sun.com/xml/jaxp/properties/schemaLanguage";
    static final String W3C_XML_SCHEMA =
    "http://www.w3.org/2001/XMLSchema";
    static final String JAXP_SCHEMA_SOURCE =
    "http://java.sun.com/xml/jaxp/properties/schemaSource";

    DOMEcho(PrintWriter out) {
        this.out = out;
    }
```

```
/**
 * Echo common attributes of a DOM2 Node and terminate output
     with an
 * EOL character.
 */
private void printlnCommon(Node n) {
    out.print(" nodeName=\"" + n.getNodeName() + "\"");

    String val = n.getNamespaceURI();
    if (val != null) {
        out.print(" uri=\"" + val + "\"");
    }

    val = n.getPrefix();
    if (val != null) {
        out.print(" pre=\"" + val + "\"");
    }

    val = n.getLocalName();
    if (val != null) {
        out.print(" local=\"" + val + "\"");
    }

    val = n.getNodeValue();
    if (val != null) {
        out.print(" nodeValue=");
        if (val.trim().equals("")) {
            // Whitespace
            out.print("[WS]");
        } else {
            out.print("\"" + n.getNodeValue() + "\"");
        }
    }
    out.println();
}

/**
 * Indent to the current level in multiples of basicIndent
 */
private void outputIndentation() {
    for (int i = 0; i < indent; i++) {
        out.print(basicIndent);
    }
}
```

```
/**
 * Recursive routine to print out DOM tree nodes
 */
private void echo(Node n) {
    // Indent to the current level before printing anything
    outputIndentation();

    int type = n.getNodeType();
    if(!noDebuggingOutput){
        switch (type) {
            case Node.ATTRIBUTE_NODE:
            out.print("ATTR:");
            printlnCommon(n);
            break;
            case Node.CDATA_SECTION_NODE:
            out.print("CDATA:");
            printlnCommon(n);
            break;
            case Node.COMMENT_NODE:
            out.print("COMM:");
            printlnCommon(n);
            break;
            case Node.DOCUMENT_FRAGMENT_NODE:
            out.print("DOC_FRAG:");
            printlnCommon(n);
            break;
            case Node.DOCUMENT_NODE:
            out.print("DOC:");
            printlnCommon(n);
            break;
            case Node.DOCUMENT_TYPE_NODE:
            out.print("DOC_TYPE:");
            printlnCommon(n);

            // Print entities if any
            NamedNodeMap nodeMap =
                ((DocumentType)n).getEntities();
            indent += 2;
            for (int i = 0; i < nodeMap.getLength(); i++) {
                Entity entity = (Entity)nodeMap.item(i);
                echo(entity);
            }
            indent -= 2;
            break;
            case Node.ELEMENT_NODE:
```

```
        out.print("ELEM:");
        printlnCommon(n);

        // Print attributes if any. Note: element attributes
            are not
        // children of ELEMENT_NODEs but are properties of
            their
        // associated ELEMENT_NODE. For this reason, they are
            printed
        // with 2x the indent level to indicate this.
        NamedNodeMap atts = n.getAttributes();
        indent += 2;
        for (int i = 0; i < atts.getLength(); i++) {
            Node att = atts.item(i);
            echo(att);
        }
        indent -= 2;
        break;
        case Node.ENTITY_NODE:
        out.print("ENT:");
        printlnCommon(n);
        break;
        case Node.ENTITY_REFERENCE_NODE:
        out.print("ENT_REF:");
        printlnCommon(n);
        break;
        case Node.NOTATION_NODE:
        out.print("NOTATION:");
        printlnCommon(n);
        break;
        case Node.PROCESSING_INSTRUCTION_NODE:
        out.print("PROC_INST:");
        printlnCommon(n);
        break;
        case Node.TEXT_NODE:
        out.print("TEXT:");
        printlnCommon(n);
        break;
        default:
        out.print("UNSUPPORTED NODE: " + type);
        printlnCommon(n);
        break;
    }
}
```

```
        // Print children if any
        indent++;
        for (Node child = n.getFirstChild(); child != null;
            child = child.getNextSibling()) {
            echo(child);
    }
    indent--;
}

private static void usage() {


    System.out.println("Please Provide the Proper Syntax");
}

public static void main(String[] args) throws Exception {
    String filename = null;


    long startTime = System.nanoTime();
        /*
        Will be calculating how long it takes to
        parse the XMl document. Start the time here.

        */

        for (int i = 0; i < args.length; i++) {

            if (args[i].startsWith("-no")) {
                noDebuggingOutput = true;
            } else {
                filename = args[i];

                // Must be last arg
                if (i != args.length - 1) {
                    usage();
                }
            }
        }

        if (filename == null) {
            usage();
        }

        // Step 1: create a DocumentBuilderFactory and configure it
    DocumentBuilderFactory dbf =
    DocumentBuilderFactory.newInstance();

    // Set namespaceAware to true to get a DOM Level 2 tree with
        nodes
    // containing namesapce information. This is necessary
        because the
    // default value from JAXP 1.0 was defined to be false.
    dbf.setNamespaceAware(true);



    // Step 2: create a DocumentBuilder that satisfies the
        constraints
    // specified by the DocumentBuilderFactory
    DocumentBuilder db = dbf.newDocumentBuilder();

    // Set an ErrorHandler before parsing
    OutputStreamWriter errorWriter = new
        OutputStreamWriter(System.err, outputEncoding);
    db.setErrorHandler(new MyErrorHandler(new
        PrintWriter(errorWriter, true)));

    // Step 3: parse the input file
    Document doc = db.parse(new File(filename));

    // Print out the DOM tree
    OutputStreamWriter outWriter =
    new OutputStreamWriter(System.out, outputEncoding);
    new DOMEcho(new PrintWriter(outWriter, true)).echo(doc);

    long estimatedTime = System.nanoTime() - startTime;
        //stop timing how long it takes after the parsing is over

    double estimatedTime2 = ((System.nanoTime() - startTime) /
        1000000000.0);
        //convert the estimatedTime to a double to displayed in
            seconds
    System.out.println("Time in nano seconds : " +
        estimatedTime);
        //display how long it took to parse in nano seconds
    System.out.printf("Time in seconds is %.9f", estimatedTime2);
        //display how long it took to parse in seconds
}
```

```
// Error handler to report errors and warnings
private static class MyErrorHandler implements ErrorHandler {
    /** Error handler output goes here */
    private PrintWriter out;

    MyErrorHandler(PrintWriter out) {
        this.out = out;
    }

    /**
     * Returns a string describing parse exception details
     */
    private String getParseExceptionInfo(SAXParseException spe) {
        String systemId = spe.getSystemId();
        if (systemId == null) {
            systemId = "null";
        }
        String info = "URI=" + systemId +
        " Line=" + spe.getLineNumber() +
        ": " + spe.getMessage();
        return info;
    }

    // The following methods are standard SAX ErrorHandler
        methods.
    // See SAX documentation for more info.

    public void warning(SAXParseException spe) throws
        SAXException {
        out.println("Warning: " + getParseExceptionInfo(spe));
    }

    public void error(SAXParseException spe) throws SAXException
        {
        String message = "Error: " + getParseExceptionInfo(spe);
        throw new SAXException(message);
    }

    public void fatalError(SAXParseException spe) throws
        SAXException {
        String message = "Fatal Error: " +
            getParseExceptionInfo(spe);
        throw new SAXException(message);
    }
}
```

```
}
```

---

---

```
SAXEcho.java

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

public class SAXEcho extends HandlerBase {

   static boolean noDebuggingOutput = false; //property to make
       sureno debugging output is created.


   public static void main (String argv [] ) {

      String filename = null; //file name of the xml file as a
          commnand line argument

      long startTime = System.nanoTime(); //start to calculate the
          time to parse the xml file

      for(int i = 0; i < argv.length; i++){

         if (argv[i].startsWith("-no")){
            noDebuggingOutput = true;
            //if the command line argument -no is given, then no
                debugging output will be displayed
         }
         else if (i == argv.length - 1){
            filename = argv[i];
            //checking if the filename is been written in as a
                command line argument
         } else if (i != argv.length -1){
            System.out.println("Usage: cmd filename");
            //if the commnand line argument has not been filled in
                correctly, then print out the statement to explain
            //what the usage should look like
         }
      }

      SAXParserFactory factory = SAXParserFactory.newInstance();
          //create a new instance of the SAX Parser Factory
      try {
         InputStream xmlInput = new FileInputStream(filename);
         //the input stream is going to be xml file. This will be in
             the command line argument
         SAXParser saxParser = factory.newSAXParser();


            DefaultHandler handler = new
                SaxHandler(noDebuggingOutput);
            //calling the default handler

            saxParser.parse(xmlInput, handler);

            //parse the xml file with the input and the handler

      } catch (Throwable err) {
         err.printStackTrace ();
      }



      long estimatedTime = System.nanoTime() - startTime;
            //stop timing how long it takes after the parsing is over

      double estimatedTime2 = ((System.nanoTime() - startTime) /
          1000000000.0);
            //convert the estimatedTime to a double to displayed in
                seconds
      System.out.println("Time in nano seconds : " + estimatedTime);
            //display how long it took to parse in nano seconds
      System.out.printf("Time in seconds is %.9f", estimatedTime2);
            //display how long it took to parse in seconds

   }
}
```

---

```
SaxHandler.java
/*
This is the source code that will print out the debugging output if
    the command line argument -no is not called
*/

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

```
public class SaxHandler extends DefaultHandler {

   boolean noDebuggingOutput = false;

   public SaxHandler(Boolean b) {

   noDebuggingOutput = b;
}


    public void startDocument() throws SAXException {

        if(!noDebuggingOutput){
        System.out.println("start document : ");
    }
    }

    public void endDocument() throws SAXException {
        if(!noDebuggingOutput){
        System.out.println("end document : ");
    }
    }

    public void startElement(String uri, String localName, String
        qName, Attributes attributes) throws SAXException {
        if(!noDebuggingOutput){
        System.out.println("start element : " + qName);
    }
    }

    public void endElement(String uri, String localName, String
        qName)
    throws SAXException {
        if(!noDebuggingOutput){
        System.out.println("end element  : " + qName);
    }
    }

    public void characters(char ch[], int start, int length) throws
        SAXException {
        if(!noDebuggingOutput){
        System.out.println("start characters : " + new String(ch,
            start, length));
    }
    }
    public void ignorableWhitespace(char ch[], int start, int
        length) throws SAXException {
    }
}
```

```
FibonacciFile.java

/*
This file will output the xml files that is 1 gb large
*/

//import statements
import java.math.BigInteger;
import java.io.*;


public class FibonacciFile {

  public static void main(String[] args) {

    BigInteger low = BigInteger.ONE;
    BigInteger high = BigInteger.ONE;

    try {
        OutputStream fout= new FileOutputStream("fibonacci.xml");
            //output the xml file of fibonacci numbers
        OutputStream bout= new BufferedOutputStream(fout);
        OutputStreamWriter out
        = new OutputStreamWriter(bout, "8859_1");

        out.write("<?xml version=\"1.0\" "); //print out the output
            stream writer
        out.write("encoding=\"ISO-8859-1\"?>\r\n");
        out.write("<Fibonacci_Numbers>\r\n");
        for (int i = 1; i <= 100000; i++) { //print out the first
            100,000 fibonacci numbers
          out.write(" <fibonacci index=\"" + i + "\">"); //print out
              the output stream writer
          out.write(low.toString());
          out.write("</fibonacci>\r\n");
          BigInteger temp = high; //calculations to calculate the
              fibonacci numbers
          high = high.add(low);
        low = temp;
      }
      out.write("</Fibonacci_Numbers>\r\n");

      out.flush(); // Don't forget to flush!
      out.close();
    }
    catch (UnsupportedEncodingException e) {
      System.out.println(
       "This VM does not support the Latin-1 character set."
       );
    }
    catch (IOException e) {
      System.out.println(e.getMessage());
    }
  }
}
```

books.xml

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer’s Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
   </book>
   <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
      <description>A former architect battles corporate zombies,
      an evil sorceress, and her own childhood to become queen
      of the world.</description>
   </book>
   <book id="bk103">
      <author>Corets, Eva</author>
      <title>Maeve Ascendant</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-11-17</publish_date>
      <description>After the collapse of a nanotechnology
      society in England, the young survivors lay the
      foundation for a new society.</description>
   </book>
   <book id="bk104">
      <author>Corets, Eva</author>
      <title>Oberon’s Legacy</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2001-03-10</publish_date>
      <description>In post-apocalypse England, the mysterious
      agent known only as Oberon helps to create a new life
      for the inhabitants of London. Sequel to Maeve
      Ascendant.</description>
   </book>
```

```
<book id="bk105">
  <author>Corets, Eva</author>
  <title>The Sundered Grail</title>
  <genre>Fantasy</genre>
  <price>5.95</price>
  <publish_date>2001-09-10</publish_date>
  <description>The two daughters of Maeve, half-sisters,
  battle one another for control of England. Sequel to
  Oberon's Legacy.</description>
</book>
<book id="bk106">
  <author>Randall, Cynthia</author>
  <title>Lover Birds</title>
  <genre>Romance</genre>
  <price>4.95</price>
  <publish_date>2000-09-02</publish_date>
  <description>When Carla meets Paul at an ornithology
  conference, tempers fly as feathers get ruffled.</description>
</book>
<book id="bk107">
  <author>Thurman, Paula</author>
  <title>Splish Splash</title>
  <genre>Romance</genre>
  <price>4.95</price>
  <publish_date>2000-11-02</publish_date>
  <description>A deep sea diver finds true love twenty
  thousand leagues beneath the sea.</description>
</book>
<book id="bk108">
  <author>Knorr, Stefan</author>
  <title>Creepy Crawlies</title>
  <genre>Horror</genre>
  <price>4.95</price>
  <publish_date>2000-12-06</publish_date>
  <description>An anthology of horror stories about roaches,
  centipedes, scorpions and other insects.</description>
</book>
<book id="bk109">
  <author>Kress, Peter</author>
  <title>Paradox Lost</title>
  <genre>Science Fiction</genre>
  <price>6.95</price>
  <publish_date>2000-11-02</publish_date>
  <description>After an inadvertant trip through a Heisenberg
  Uncertainty Device, James Salway discovers the problems
```

```
     of being quantum.</description>
   </book>
   <book id="bk110">
     <author>O'Brien, Tim</author>
     <title>Microsoft .NET: The Programming Bible</title>
     <genre>Computer</genre>
     <price>36.95</price>
     <publish_date>2000-12-09</publish_date>
     <description>Microsoft's .NET initiative is explored in
     detail in this deep programmer's reference.</description>
   </book>
   <book id="bk111">
     <author>O'Brien, Tim</author>
     <title>MSXML3: A Comprehensive Guide</title>
     <genre>Computer</genre>
     <price>36.95</price>
     <publish_date>2000-12-01</publish_date>
     <description>The Microsoft MSXML3 parser is covered in
     detail, with attention to XML DOM interfaces, XSLT processing,
     SAX and more.</description>
   </book>
   <book id="bk112">
     <author>Galos, Mike</author>
     <title>Visual Studio 7: A Comprehensive Guide</title>
     <genre>Computer</genre>
     <price>49.95</price>
     <publish_date>2001-04-16</publish_date>
     <description>Microsoft Visual Studio 7 is explored in depth,
     looking at how Visual Basic, Visual C++, C#, and ASP+ are
     integrated into a comprehensive development
     environment.</description>
   </book>
</catalog>
```

Snippet of ActorPreludeSample.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 3.0
    Partwise//EN" "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.0">
  <movement-title>Prelude to a Tragedy</movement-title>
  <identification>
    <creator type="composer">Lee Actor</creator>
    <rights>  2004 Polygames.   All Rights Reserved.</rights>
    <encoding>
      <software>Finale 2011 for Windows</software>
      <software>Dolet 6.0 for Finale</software>
      <encoding-date>2011-08-08</encoding-date>
      <supports attribute="new-system" element="print" type="yes"
          value="yes"/>
      <supports attribute="new-page" element="print" type="yes"
          value="yes"/>
    </encoding>
  </identification>
  <defaults>
    <scaling>
      <millimeters>3.9956</millimeters>
      <tenths>40</tenths>
    </scaling>
    <page-layout>
      <page-height>3560</page-height>
      <page-width>2797</page-width>
      <page-margins type="even">
        <left-margin>141</left-margin>
        <right-margin>115</right-margin>
        <top-margin>127</top-margin>
        <bottom-margin>45</bottom-margin>
      </page-margins>
      <page-margins type="odd">
        <left-margin>177</left-margin>
        <right-margin>79</right-margin>
        <top-margin>127</top-margin>
        <bottom-margin>45</bottom-margin>
      </page-margins>
    </page-layout>
    <system-layout>
      <system-margins>
        <left-margin>0</left-margin>
        <right-margin>0</right-margin>
      </system-margins>
      <system-distance>39</system-distance>
      <top-system-distance>39</top-system-distance>
      <system-dividers>
        <left-divider print-object="yes"/>
        <right-divider print-object="no"/>
      </system-dividers>
    </system-layout>
    <staff-layout>
      <staff-distance>93</staff-distance>
    </staff-layout>
    <appearance>
      <line-width type="stem">0.957</line-width>
      <line-width type="beam">5.2083</line-width>
      <line-width type="staff">0.957</line-width>
      <line-width type="light barline">1.875</line-width>
      <line-width type="heavy barline">5.0391</line-width>
      <line-width type="leger">1.875</line-width>
      <line-width type="ending">0.957</line-width>
      <line-width type="wedge">1.875</line-width>
      <line-width type="enclosure">0.957</line-width>
      <line-width type="tuplet bracket">0.957</line-width>
      <note-size type="grace">60</note-size>
      <note-size type="cue">60</note-size>
      <distance type="hyphen">60</distance>
      <distance type="beam">8</distance>
    </appearance>
    <music-font font-family="Maestro,engraved" font-size="11.3"/>
    <word-font font-family="Times New Roman" font-size="5.7"/>
  </defaults>
  <credit page="1">
    <credit-type>title</credit-type>
    <credit-words default-x="1447" default-y="3477"
        font-size="19.5" justify="center" valign="top">Prelude to a
        Tragedy</credit-words>
  </credit>
  <credit page="1">
    <credit-type>composer</credit-type>
    <credit-words default-x="2718" default-y="3387" font-size="7.8"
        justify="right" valign="top">Lee Actor (2003)</credit-words>
  </credit>
  <credit page="1">
    <credit-type>rights</credit-type>
```

```
    <credit-words default-x="1447" default-y="45" font-size="7.8"
        justify="center" valign="bottom" xml:space="preserve"> 2004
        Polygames. All Rights Reserved.</credit-words>
  </credit>
  <credit page="2">
    <credit-type>page number</credit-type>
    <credit-words default-x="1412" default-y="45" font-size="7.8"
        halign="center" valign="bottom">- 2 -</credit-words>
  </credit>
  <credit page="3">
    <credit-type>page number</credit-type>
    <credit-words default-x="1447" default-y="45" font-size="7.8"
        halign="center" valign="bottom">- 3 -</credit-words>
  </credit>
  <credit page="4">
    <credit-type>page number</credit-type>
    <credit-words default-x="1412" default-y="45" font-size="7.8"
        halign="center" valign="bottom">- 4 -</credit-words>
  </credit>
  <part-list>
    <part-group number="1" type="start">
      <group-symbol default-x="-7">bracket</group-symbol>
      <group-barline>yes</group-barline>
    </part-group>
    <score-part id="P1">
      <part-name>Piccolo</part-name>
      <part-abbreviation>Picc.</part-abbreviation>
      <score-instrument id="P1-I18">
        <instrument-name>Picc. (V2k)</instrument-name>
      </score-instrument>
      <midi-instrument id="P1-I18">
        <midi-channel>1</midi-channel>
        <midi-program>73</midi-program>
        <volume>80</volume>
        <pan>0</pan>
      </midi-instrument>
    </score-part>
    <part-group number="2" type="start">
      <group-name>1
2</group-name>
      <group-barline>yes</group-barline>
    </part-group>
    <score-part id="P2">
      <part-name>Flutes</part-name>
      <part-abbreviation>Fl.</part-abbreviation>
      <score-instrument id="P2-I19">
        <instrument-name>Fl. (V2k)</instrument-name>
      </score-instrument>
      <midi-instrument id="P2-I19">
        <midi-channel>2</midi-channel>
        <midi-program>74</midi-program>
        <volume>80</volume>
        <pan>0</pan>
      </midi-instrument>
    </score-part>
    <part-group number="2" type="stop"/>
    <part-group number="2" type="start">
      <group-name>1
2</group-name>
      <group-barline>yes</group-barline>
    </part-group>
    <score-part id="P3">
      <part-name>Oboes</part-name>
      <part-abbreviation>Ob.</part-abbreviation>
      <score-instrument id="P3-I2">
        <instrument-name>Ob. (V2k)</instrument-name>
      </score-instrument>
      <midi-instrument id="P3-I2">
        <midi-channel>3</midi-channel>
        <midi-program>69</midi-program>
        <volume>80</volume>
        <pan>0</pan>
      </midi-instrument>
    </score-part>
    <part-group number="2" type="stop"/>
    <score-part id="P4">
      <part-name>English Horn</part-name>
      <part-abbreviation>E. H.</part-abbreviation>
      <score-instrument id="P4-I11">
        <instrument-name>E.H. (V2k)</instrument-name>
      </score-instrument>
      <midi-instrument id="P4-I11">
        <midi-channel>4</midi-channel>
        <midi-program>70</midi-program>
        <volume>80</volume>
        <pan>0</pan>
      </midi-instrument>
    </score-part>
    <part-group number="2" type="start">
      <group-name>1
```

```
Snippet of EPAXMLDownload.xml

?xml version="1.0" encoding="UTF-8"?>
<Document xsi:noNamespaceSchemaLocation="EPA_GEODATA_v1.0.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Header>
<Title>USEPA Geospatial Data</Title>
<Creator>Environmental Protection Agency</Creator>
<Subject>USEPA Geospatial Data</Subject>
<Description>This XML file was produced by US EPA and contains data
    specifying the locations of EPA regulated facilities or
    cleanups that are being provided by EPA for use by commercial
    mapping services and others with an interest in using this
    information. Updates to this file are produced on a regular
    basis by EPA and those updates as well as documentation
    describing the contents of the file can be found at
    URL:http://www.epa.gov/enviro</Description>
<Date>NOV-09-2014</Date>
</Header>
<FacilitySite registryId="110007915364">
<FacilitySiteName>GREAT SOUTHERN WOOD PRESERVING
    INC</FacilitySiteName>
<LocationAddressText>1100 HIGHWAY 431 NORTH</LocationAddressText>
<LocalityName>ABBEVILLE</LocalityName>
<LocationAddressStateCode>AL</LocationAddressStateCode>
<LocationZIPCode>36310</LocationZIPCode>
<LatitudeMeasure>31.624667</LatitudeMeasure>
<LongitudeMeasure>-85.272156</LongitudeMeasure>
<HorizontalCoordinateReferenceSystemDatumName>NAD83</HorizontalCoordinateReferenceSystemDatum
<HorizontalCollectionMethodName>ADDRESS MATCHING-BLOCK
    FACE</HorizontalCollectionMethodName>
<GeneralProfileElectronicAddress>
<ElectronicAddressText>http://oaspub.epa.gov/enviro/fac_gateway.main?p_regid=110007915364</El
<ElectronicAddressTypeName>URL</ElectronicAddressTypeName>
</GeneralProfileElectronicAddress>
<Program>
<ProgramCommonName>TRIS</ProgramCommonName>
<ProgramAcronymName>TRIS</ProgramAcronymName>
<ProgramIdentifier>36310GRTSTHWY43</ProgramIdentifier>
<ProgramInterestType>TRI REPORTER</ProgramInterestType>
<ProgramFullName>Toxics Release Inventory Program</ProgramFullName>
<ProgramDescription>The Toxics Release Inventory (TRI) is a
    publicly available EPA database that contains information on
    toxic chemical releases and other waste management activities
    reported annually by certain covered industry groups as well as
    federal facilities.</ProgramDescription>
<ProgramFacilitySiteName>GREAT SOUTHERN WOOD PRESERVING
    INC</ProgramFacilitySiteName>
<ProgramProfileElectronicAddress>
<ElectronicAddressText>http://www.epa.gov/tri/</ElectronicAddressText>
<ElectronicAddressTypeName>URL</ElectronicAddressTypeName>
</ProgramProfileElectronicAddress>
</Program>
</FacilitySite>
<FacilitySite registryId="110000369084">
<FacilitySiteName>REMBRANDT FOODS- ABBEVILLE</FacilitySiteName>
<LocationAddressText>496 INDUSTRIAL PARK RD</LocationAddressText>
<LocalityName>ABBEVILLE</LocalityName>
<LocationAddressStateCode>AL</LocationAddressStateCode>
<LocationZIPCode>36310</LocationZIPCode>
<LatitudeMeasure>31.555725</LatitudeMeasure>
<LongitudeMeasure>-85.287302</LongitudeMeasure>
<HorizontalCoordinateReferenceSystemDatumName>NAD83</HorizontalCoordinateReferenceSystemDatum
<HorizontalCollectionMethodName>INTERPOLATION - DIGITAL MAP SRCE
    (TIGER)</HorizontalCollectionMethodName>
<GeneralProfileElectronicAddress>
<ElectronicAddressText>http://oaspub.epa.gov/enviro/fac_gateway.main?p_regid=110000369084</El
<ElectronicAddressTypeName>URL</ElectronicAddressTypeName>
</GeneralProfileElectronicAddress>
<Program>
<ProgramCommonName>TRIS</ProgramCommonName>
<ProgramAcronymName>TRIS</ProgramAcronymName>
<ProgramIdentifier>36310CTLRGINDUS</ProgramIdentifier>
<ProgramInterestType>TRI REPORTER</ProgramInterestType>
<ProgramFullName>Toxics Release Inventory Program</ProgramFullName>
<ProgramDescription>The Toxics Release Inventory (TRI) is a
    publicly available EPA database that contains information on
    toxic chemical releases and other waste management activities
    reported annually by certain covered industry groups as well as
    federal facilities.</ProgramDescription>
<ProgramFacilitySiteName>REMBRANDT FOODS-
    ABBEVILLE</ProgramFacilitySiteName>
<ProgramProfileElectronicAddress>
<ElectronicAddressText>http://www.epa.gov/tri/</ElectronicAddressText>
<ElectronicAddressTypeName>URL</ElectronicAddressTypeName>
</ProgramProfileElectronicAddress>
</Program>
</FacilitySite>
<FacilitySite registryId="110038256650">
```

```
<FacilitySiteName>TRAWICK HDWE INC</FacilitySiteName>
<LocationAddressText>112 S DOSWELL ST</LocationAddressText>
<LocalityName>ABBEVILLE</LocalityName>
<LocationAddressStateCode>AL</LocationAddressStateCode>
<LocationZIPCode>36310</LocationZIPCode>
<LatitudeMeasure>31.570712</LatitudeMeasure>
<LongitudeMeasure>-85.249449</LongitudeMeasure>
<HorizontalCoordinateReferenceSystemDatumName>NAD83</HorizontalCoordinateReferenceSystemDatum
<HorizontalCollectionMethodName>ADDRESS MATCHING-HOUSE
    NUMBER</HorizontalCollectionMethodName>
<GeneralProfileElectronicAddress>
<ElectronicAddressText>http://oaspub.epa.gov/enviro/fac_gateway.main?p_regid=110038256650</El
<ElectronicAddressTypeName>URL</ElectronicAddressTypeName>
</GeneralProfileElectronicAddress>
</FacilitySite>
<FacilitySite registryId="110025379347">
<FacilitySiteName>C AND B PIPING INC</FacilitySiteName>
<LocationAddressText>4128 ADAMSVILLE PARKWAY</LocationAddressText>
<LocalityName>ADAMSVILLE</LocalityName>
<LocationAddressStateCode>AL</LocationAddressStateCode>
<LocationZIPCode>35005-1370</LocationZIPCode>
<LatitudeMeasure>33.597728</LatitudeMeasure>
<LongitudeMeasure>-86.943685</LongitudeMeasure>
<HorizontalCoordinateReferenceSystemDatumName>NAD83</HorizontalCoordinateReferenceSystemDatum
<HorizontalCollectionMethodName>ADDRESS MATCHING-HOUSE
    NUMBER</HorizontalCollectionMethodName>
<GeneralProfileElectronicAddress>
<ElectronicAddressText>http://oaspub.epa.gov/enviro/fac_gateway.main?p_regid=110025379347</El
<ElectronicAddressTypeName>URL</ElectronicAddressTypeName>
</GeneralProfileElectronicAddress>
<Program>
<ProgramCommonName>AIRS/AFS AIR MAJOR</ProgramCommonName>
<ProgramAcronymName>AIRS/AFS AIR MAJOR</ProgramAcronymName>
<ProgramIdentifier>0107300489</ProgramIdentifier>
<ProgramInterestType>AIR MAJOR</ProgramInterestType>
<ProgramFullName>Aerometric Information Retrieval System/Airs
    Facility Subsystem Air Major</ProgramFullName>
<ProgramDescription>A clean air act stationary source major
    discharger of air pollutants according to the Alabama power
    decision's definition of a major source or the 1993 EPA
    compliance monitoring branch classification guidance. A
    facility is classified as a major discharger if: (a) actual or
    potential emissions are above the applicable major source
    thresholds; or (b) actual or potential controlled emissions
    more than 100 tons/year as per Alabama power decision; or (c)
    unregulated pollutant actual or potential controlled emissions
    more than 100 tons/year as per Alabama power
    decision.</ProgramDescription>
<ProgramFacilitySiteName>C &amp; B PIPING,INC.
    ADAMSVILLE</ProgramFacilitySiteName>
<ProgramProfileElectronicAddress>
<ElectronicAddressText>http://www.epa.gov/air/oaqps/permits/obtain.html</ElectronicAddressTex
<ElectronicAddressTypeName>URL</ElectronicAddressTypeName>
</ProgramProfileElectronicAddress>
</Program>
```

Snippet of fibonacci.xml

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<Fibonacci_Numbers>
  <fibonacci index="1">1</fibonacci>
  <fibonacci index="2">1</fibonacci>
  <fibonacci index="3">2</fibonacci>
  <fibonacci index="4">3</fibonacci>
  <fibonacci index="5">5</fibonacci>
  <fibonacci index="6">8</fibonacci>
  <fibonacci index="7">13</fibonacci>
  <fibonacci index="8">21</fibonacci>
  <fibonacci index="9">34</fibonacci>
  <fibonacci index="10">55</fibonacci>
  <fibonacci index="11">89</fibonacci>
  <fibonacci index="12">144</fibonacci>
  <fibonacci index="13">233</fibonacci>
  <fibonacci index="14">377</fibonacci>
  <fibonacci index="15">610</fibonacci>
  <fibonacci index="16">987</fibonacci>
  <fibonacci index="17">1597</fibonacci>
  <fibonacci index="18">2584</fibonacci>
  <fibonacci index="19">4181</fibonacci>
  <fibonacci index="20">6765</fibonacci>
  <fibonacci index="21">10946</fibonacci>
  <fibonacci index="22">17711</fibonacci>
  <fibonacci index="23">28657</fibonacci>
  <fibonacci index="24">46368</fibonacci>
  <fibonacci index="25">75025</fibonacci>
  <fibonacci index="26">121393</fibonacci>
  <fibonacci index="27">196418</fibonacci>
  <fibonacci index="28">317811</fibonacci>
  <fibonacci index="29">514229</fibonacci>
  <fibonacci index="30">832040</fibonacci>
  <fibonacci index="31">1346269</fibonacci>
  <fibonacci index="32">2178309</fibonacci>
  <fibonacci index="33">3524578</fibonacci>
  <fibonacci index="34">5702887</fibonacci>
  <fibonacci index="35">9227465</fibonacci>
  <fibonacci index="36">14930352</fibonacci>
  <fibonacci index="37">24157817</fibonacci>
  <fibonacci index="38">39088169</fibonacci>
  <fibonacci index="39">63245986</fibonacci>
  <fibonacci index="40">102334155</fibonacci>
```

```
<fibonacci index="41">165580141</fibonacci>
<fibonacci index="42">267914296</fibonacci>
<fibonacci index="43">433494437</fibonacci>
<fibonacci index="44">701408733</fibonacci>
<fibonacci index="45">1134903170</fibonacci>
<fibonacci index="46">1836311903</fibonacci>
<fibonacci index="47">2971215073</fibonacci>
<fibonacci index="48">4807526976</fibonacci>
<fibonacci index="49">7778742049</fibonacci>
<fibonacci index="50">12586269025</fibonacci>
<fibonacci index="51">20365011074</fibonacci>
<fibonacci index="52">32951280099</fibonacci>
<fibonacci index="53">53316291173</fibonacci>
<fibonacci index="54">86267571272</fibonacci>
<fibonacci index="55">139583862445</fibonacci>
<fibonacci index="56">225851433717</fibonacci>
<fibonacci index="57">365435296162</fibonacci>
<fibonacci index="58">591286729879</fibonacci>
<fibonacci index="59">956722026041</fibonacci>
<fibonacci index="60">1548008755920</fibonacci>
<fibonacci index="61">2504730781961</fibonacci>
<fibonacci index="62">4052739537881</fibonacci>
<fibonacci index="63">6557470319842</fibonacci>
<fibonacci index="64">10610209857723</fibonacci>
<fibonacci index="65">17167680177565</fibonacci>
<fibonacci index="66">27777890035288</fibonacci>
<fibonacci index="67">44945570212853</fibonacci>
<fibonacci index="68">72723460248141</fibonacci>
<fibonacci index="69">117669030460994</fibonacci>
<fibonacci index="70">190392490709135</fibonacci>
<fibonacci index="71">308061521170129</fibonacci>
<fibonacci index="72">498454011879264</fibonacci>
<fibonacci index="73">806515533049393</fibonacci>
<fibonacci index="74">1304969544928657</fibonacci>
<fibonacci index="75">2111485077978050</fibonacci>
<fibonacci index="76">3416454622906707</fibonacci>
<fibonacci index="77">5527939700884757</fibonacci>
<fibonacci index="78">8944394323791464</fibonacci>
<fibonacci index="79">14472334024676221</fibonacci>
<fibonacci index="80">23416728348467685</fibonacci>
<fibonacci index="81">37889062373143906</fibonacci>
<fibonacci index="82">61305790721611591</fibonacci>
<fibonacci index="83">99194853094755497</fibonacci>
<fibonacci index="84">160500643816367088</fibonacci>
<fibonacci index="85">259695496911122585</fibonacci>
```

```
<fibonacci index="86">420196140727489673</fibonacci>
<fibonacci index="87">679891637638612258</fibonacci>
<fibonacci index="88">1100087778366101931</fibonacci>
<fibonacci index="89">1779979416004714189</fibonacci>
<fibonacci index="90">2880067194370816120</fibonacci>
<fibonacci index="91">4660046610375530309</fibonacci>
<fibonacci index="92">7540113804746346429</fibonacci>
<fibonacci index="93">12200160415121876738</fibonacci>
<fibonacci index="94">19740274219868223167</fibonacci>
<fibonacci index="95">31940434634990099905</fibonacci>
<fibonacci index="96">51680708854858323072</fibonacci>
<fibonacci index="97">83621143489848422977</fibonacci>
<fibonacci index="98">135301852344706746049</fibonacci>
<fibonacci index="99">218922995834555169026</fibonacci>
<fibonacci index="100">354224848179261915075</fibonacci>
```