


STANFORD UNIVERSITY

CS193J: Programming in Java
Winter Quarter 2003

Lecture 12
Files and Streams, XML, SAX XML Parsing

Manu Kumar
sneaker@stanford.edu

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar




STANFORD UNIVERSITY

Handouts

- 2 Handouts for today!
 - #27: XML
 - #28: SAX XML Parsing

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar




STANFORD UNIVERSITY

Recap

- Last Time
 - More HW3b intuition...
 - MVC
 - Model View Controller paradigm
 - JTable
 - Exceptions
 - try/catch/finally
 - Exception patterns
- Assigned Work Reminder
 - HW 3a: ThreadBank
 - HW 3b: LinkTester
 - Both due before midnight on Wednesday, August 6th, 2003

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar




STANFORD UNIVERSITY

Today

- Today:
 - Files and Streams
 - XML
 - Introduction
 - Java XML
 - DOM
 - DotPanel example
 - XML Scenarios
 - SAX XML Parsing (potentially next time)

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar




STANFORD UNIVERSITY

Files and Streams (Handout #26)

- File
 - Represents a file or directory
 - Platform independent way to test file attributes, list directories
 - Java abstracts away the ugliness of dealing with files quite nicely
 - Do not open File object directly, instead we use streams...
- Streams
 - Way to deal with input and output
 - A useful abstraction...

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar



STANFORD UNIVERSITY

Streams!??

- Water analogy
 - Think of streams as pipes for water
 - Do you know whether the water that comes out of your tap is coming from a) the ocean b) some river c) a water tank d) a water buffalo?
- Idea:
 - You abstract away what the stream is connected to and perform all your I/O operations on the stream
 - The stream may be connected to a file on a floppy, a file on a hard disk, a network connection or may even just be in memory!

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar



STANFORD UNIVERSITY InputStream / OutputStream

- Base class streams
 - Very few features beyond read() and write()
- Deal with plain bytes
- Usually used through an intermediate class or a subclass
 - We'll see this in a bit

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar



STANFORD UNIVERSITY Types of Streams

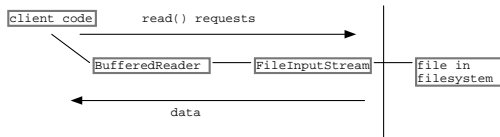
- InputStream / OutputStream
 - Base class streams with few features
 - read() and write()
- FileInputStream / FileOutputStream
 - Specifically for connecting to files
- ByteArrayInputStream / ByteArrayOutputStream
 - Use an in-memory array of bytes for storage!
- BufferedInputStream / BufferedOutputStream
 - Improve performance by adding buffers
 - Should almost always use buffers
- BufferedReader / BufferedWriter
 - Convert bytes to unicode Char and String data
 - Probably most useful for what we need

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar



STANFORD UNIVERSITY Common Use Scenario

- Using a BufferedReader to read data from a file...

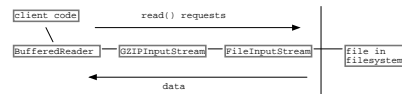


Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar



STANFORD UNIVERSITY Hierarchy of Streams

- Java provides a hierarchy of streams
 - Think of this as different “filters” you can add on to your water pipe
 - Some may compress/decompress data
 - Some may provide buffers
- Common Use Scenario
 - Streams are used by layering them together to form the type of “pipe” we eventually want



Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar



STANFORD UNIVERSITY Streams and Threads

- When a thread sends a read() to a stream, if the data is not ready, the thread blocks in the call to read(). When the data is there, the thread unblocks and the call to read() returns
- The reading/writing code does not need to do anything special
- Read 10 things at once – create 10 threads!

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar



STANFORD UNIVERSITY Reading Example

```

public void readLines(String fname) {
    try {
        // Build a reader on the fname, (also works with File object)
        BufferedReader in = new BufferedReader(new
            FileReader(fname));
        String line;
        while ((line = in.readLine()) != null) {
            // do something with 'line'
            System.out.println(line);
        }
        in.close(); // polite
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}
  
```

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar



Writing Example

```
public void writeLines(String fname) {
    try {
        // Build a writer on the fname (also works on File objects)
        BufferedWriter out = new BufferedWriter(new FileWriter(fname));

        // Send out.print(), out.println() to write chars
        for (int i=0; i<data.size(); i++) {
            out.println( ... ith data string ... );
        }

        out.close();           // polite
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}
```



HTTP

- Java has build-in and very elegant support for HTTP
- Code on the handout is what you will need for HW #3 Part b!
- URL
 - Uniform Resource Location
 - <http://cs193j.stanford.edu>
- URLConnection
 - To open a network connection to a URL and be able to get a stream from it to read data!



HTTP Example

```
public static void dumpURL(String urlString) {
    try {
        URL url = new URL(urlString);
        URLConnection conn = url.openConnection();
        InputStream stream = conn.getInputStream();
        BufferedReader in = new BufferedReader( new
        InputStreamReader(stream));

        String line;
        while ( (line = in.readLine()) != null) {
            System.out.println(line);
        }
        in.close();
    }
    catch (MalformedURLException e) {
        e.printStackTrace();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}
```



Reading in larger chunks...

```
/*
 * Given a reader, reads all its chars into a StringBuffer.
 */
public static StringBuffer readIntoBuffer(Reader in) throws IOException {
    // char array for temporary storage
    char[] chars = new char[512];
    int len;

    StringBuffer buff = new StringBuffer();

    // call read() to put chars into the array
    // read() returns -1 on EOF
    while ((len = in.read(chars, 0, chars.length)) >= 0) {
        // append the chars into the String Buffer
        buff.append(chars, 0, len);
    }
    return(buff);
}
```



XML (Handout #27)

- eXtensible Markup Language
 - Textual data format
 - A way of describing bindings using text
 - Simple
 - Lost of hype for something so simple
 - Standardized
 - A data exchange format
 - Helps with the basic problem of structure and parsing
 - Applications still need to agree on the meaning of the data



DTD

- Data Type Definition
 - Formal description of the allowed structure for a class of XML elements
 - A parser or other tool can formally check that a document meets the DTD structure
- XML resources
 - <http://www.xml.org>
 - <http://www.w3.org/XML>
 - <http://java.sun.com/xml>



XML Tags

- Tags
 - Meta content in text
 - Similar to HTML tags
 - Here is some text `<red>with this</red>` marked as red
 - Tags are case-sensitive, unlike HTML
 - May contain raw text or other tags
 - `<tag></tag>` is equivalent to `<tag />`
- Tag Attributes
 - Store name value binding inside a tag
 - May use single quote or double quote
 - `<dot x="72" y="13" />`



Special Characters

- Some characters are used as part of the description and therefore must be encoded
- All end with a ;
- Examples
 - `<` encoded as `<`;
 - `>` encoded as `>`;
 - `&` encode as `&`;
 - `"` encode as `"`;
 - `'` encode as `'`;



XML Strategies

- Text form
 - Used like HTML, lots of text with tags sprinkled in between
 - `<foo>And here is some text</foo>`
- Tree form
 - Written as a tree structure...
 - `<person>`
 - `<name>Hans Gruber</name>`
 - `<id>123456</id>`
 - `<username>hans</username>`
 - `</person>`
 - More commonly used for XML



Tags vs. Attributes

- The following are equivalent
 - Attribute Method
 - `<dot x="27" y="13">`
 - Tag Method
 - `<dot>`
 - `<x>27</x>`
 - `<y>13</y>`
 - `</dot>`



Tags vs. Attributes Style

- Tags and attributes can encode equivalent information
- Rules of thumb
 - Use attribute method when the data is short
 - `<dot x="6" y="13" />`
 - Use the tag method if the data is lengthy
 - `<description>How did our constructed suburban landscape come to be so unpleasant, and what to do about it. The Geography of Nowhere is a landmark work in growth of the New Urbanism movement.</description>`
 - Use the tag way if a node can have an arbitrary number of child nodes
 - `<parent> <child>..</child> <child>..</child> <child>..</child> </parent>`



Dots XML example

- Dots – a set of (x,y) points
 - Root node: "dots"
 - Child nodes: "dot" with x and y attributes
- ```
<?xml version="1.0" encoding="UTF-8"?>
<dots>
 <dot x="72" y="101" />
 <dot x="170" y="164" />
 <dot x="184" y="158" />
 <dot x="194" y="146" />
 <dot x="191" y="133" />
 <dot x="164" y="84" />
 <dot x="119" y="89" />
</dots>
```



- JAXP project
  - <http://java.sun.com/xml>
  - Java API for XML Processing (JAXP)
  - Supports processing of XML documents using DOM, SAX, and XSLT.
  - Enables applications to parse and transform XML documents independent of a particular XML processing implementation
- SAX
  - <http://www.saxproject.org/>
  - Simple API for XML
  - First widely adopted API for XML in Java



- For Java 1.3
  - jaxp.jar and crimson.jar are required
- For Java 1.4
  - XML classes are part of the default distribution
    - No additional jar files needed
  - Use 1.4!



- Document Object Model
  - Tree of XML nodes
  - Can iterate over the tree to look at the nodes
  - Can edit the tree to add/remove nodes
- DOM Document
  - In memory representation of the entire tree
  - Has a pointer to the root node
  - Building the DOM tree is expensive
    - Each node is a java object



- Represents each <tag>... </tag> section
- Nodes contain other children nodes
- Nodes can have attribute/value bindings
- There can be free form text between the nodes
  - These usually show up as Text Nodes
- In JAXP, Element is a subclass of Node
  - Our code will tend to use Element, since it responds to getAttribute/setAttribute
- Root Node
  - Contains all the content and is the one child of the document object



- Our approach
  - Use the DocumentBuilder.parse() method to read the XML and build the DOM in memory
  - Traverse it to examine the nodes and get the data out
- Alternatives
  - SAX – shows the nodes of the XML document one at a time; Does not build the tree in memory
  - Use the DOM tree as our data model itself
    - No translation step for reading or writing



- Imports for XML support
 

```
// Standard imports for XML
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import org.w3c.dom.*;
....
```



STANFORD UNIVERSITY  
Read DOM into Memory

```
// The following is the standard incantation to get a Document object
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

dbf.setValidating(false);

DocumentBuilder db = null;
try {
 db = dbf.newDocumentBuilder();
} catch (ParserConfigurationException pce) {
 pce.printStackTrace();
}

// Parse the XML to build the whole doc tree
Document doc = db.parse(file);
```



STANFORD UNIVERSITY  
Traversal Methods

```
// Get root node of document
Element root = doc.getDocumentElement();

// Get list of children of given tag name
NodeList list = root.getElementsByTagName("tagname");

// Number of children in list
int len = list.getLength();

// Get nth child
Element elem = (Element) list.item(n);

// Get an attribute out of a element
// (returns "" if there is no such attribute)
String s = elem.getAttribute("attribute");
```



STANFORD UNIVERSITY  
Writing an XML file

- Our approach
  - Construct the DOM Document tree in memory
  - Trick: Downcast the Document object into an XMLDocument
    - XMLDocument responds to a write() message where it writes itself out in text form!
- Alternatives
  - Using XSLT (complicated)
  - Faster to write using println()
    - But then we have to manually take care of the tags and encoding



STANFORD UNIVERSITY  
DOM Writing code

- Two line trick for writing out the DOM...
- ```
// 1. Cast the doc down to an XmlDocument
XmlDocument x = (XmlDocument) doc;

// 2. XmlDocument knows how to write itself out Woo Hoo!
x.write(out, "UTF-8");
```



STANFORD UNIVERSITY
DOM Editing Methods

```
// Create a new node (still needs to be added)
Element elem = document.createElement("tagname");

// Append a child node to an existing node
node.appendChild(elem);

// Set an attribute/value binding in a node.
// (the strings should be xml-ready text --
// no embedded " or < or &)
elem.setAttribute(attr-string, value-string);
```



STANFORD UNIVERSITY
Dots Example

- Build on previous DotPanel example...
 - Mouse tracking
 - clicking makes a new point, clicking on an existing point moves it
 - Smart repaint
 - only repaints the needed rectangle when a dot moves
 - File Open/Save
 - uses the KDeskFrame/KinnerFrame code to provide a document/window interface
 - Serialization
 - has code to save and load the data model using Java serialization. See saveSerial() and loadSerial()
 - XML
 - has code that uses the Java XML package (JAXP-1.1) to save and load the data model to XML text. See saveXML() and loadXML().
 - <http://java.sun.com/xml/>
 - The Jaxp libraries are in jaxp.jar and crimson.jar for Java 1.3 and earlier

STANFORD UNIVERSITY

Dots Example...

- Code walkthrough of selected sections...
 - In emacs...

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar

STANFORD UNIVERSITY

XML Takes

- Standard format
 - Similar to the plain text file
 - Provides a lowest common denominator approach easy for other programs to parse
 - Examples
 - Data files, preferences files, data exchange format
- Big and Slow
 - XML-bloat
 - Text description of data usually takes more space!
 - Tradeoff between compatibility and saving programmer time and space
 - Space is getting cheaper (network bandwidth and hard disk)

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar

STANFORD UNIVERSITY

XSL / XSLT

- A movement to keep “presentation” out of XML
- XSL – eXtensible Stylesheet Language
 - Like HTML style sheets for XML
- XSLT – XSL Transformations
 - Used to translate from one XML format into another
 - For example to take XML data and format it as HTML

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar

STANFORD UNIVERSITY

Backward / Forward Compatibility

- Tag names declare what each piece of data is
 - Makes it easier to have additional information in the format to ensure backward/forward compatibility
- Backward Compatibility
 - A new version of the application will be able to read the documents from the old version
- Forward Compatibility
 - An old version of the application will be able to read documents from the new version
- Roundtrip Compatibility
 - New and old version can read each other and we can move between versions transparently

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar

STANFORD UNIVERSITY

SAX XML Parsing (Handout #28)

- SAX parsing is cheaper than DOM parsing
 - SAX tells you of each element as it is found in a single pass of the XML document
 - We must maintain state ourselves
- XMLDotReader Examples
 - Code walkthrough in emacs...

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar

STANFORD UNIVERSITY

Summary!

- Today
 - Files and Streams
 - XML
 - Introduction
 - Java XML
 - DOM
 - DotPanel example
 - XML Scenarios
 - SAX XML Parsing (?)
- Assigned Work Reminder
 - HW 3a: ThreadBank
 - HW 3b: LinkTester
 - Both due before midnight on Wednesday, August 6th, 2003

Tuesday, July 29th, 2003 Copyright © 2003, Manu Kumar