

XML: Introduction to XML Parsing

Qiang Yang, SFU

Thanks: Ethan Cerami
New York University

2/1/01 XML Parsing 1

Road Map

- What is a Parser?
 - Defining Parser Responsibilities
 - Evaluating Parsers
- Validation
 - Validating v. Nonvalidating Parsers
- XML Interfaces
 - Object v. Tree Based Interfaces
 - Interface Standards: DOM, SAX
- Java XML Parsers

2/1/01 XML Parsing 2

What is an XML Parser?

2/1/01 XML Parsing 3

The Big Picture

```

    graph LR
      XMLDoc[XML Document] --> XMLParser[XML Parser]
      XMLParser --> JavaApp[Java Application Or Servlet]
  
```

An XML Parser enables your Java application or Servlet to more easily access XML Data.

2/1/01 XML Parsing 4

Defining Parser Responsibilities

A XML Parser has three main responsibilities:

- Retrieve and Read an XML document
 - For example, the file may reside on the local file system or on another web site.
 - The parser takes care of all the necessary network connections and/or file connections.
 - This helps simplify your work, as you do not need to worry about creating your own network connections.

2/1/01 XML Parsing 5

Parser Responsibilities

- Ensure that the document adheres to specific standards.
 - Does the document match the DTD?
 - Is the document well-formed?
- Make the document contents available to your application.
 - The parser will parse the XML document, and make this data available to your application.

2/1/01 XML Parsing 6

Why use an XML Parser?

- If your application is going to use XML, you could write your own parser.
- But, it makes more sense to use a pre-built XML parser.
- This enables you to do build your application much more quickly.

2/1/01

XML Parsing

7

Evaluating Parsers

2/1/01

XML Parsing

8

Questions to ask

- When evaluating which XML Parser to use, there are two very important questions to ask:
 - Is the Parser validating or non-validating?
 - What interface does the parser provide to the XML document?
- We will explore each of these question in detail...

2/1/01

XML Parsing

9

XML Validation

2/1/01

XML Parsing

10

XML Validation

- **Validating Parser**
 - a parser that verifies that the XML document adheres to the DTD.
- **Non-Validating Parser**
 - a parser that does not check the DTD.
- Lots of parsers provide an option to turn validation on or off.

2/1/01

XML Parsing

11

Performance and Memory

- **Questions:**
 - Which parser will have better performance?
 - Which parser will take up less memory?
- **Validating parsers:**
 - more useful
 - slower
 - take up more memory
- **Non-validating parsers:**
 - less useful
 - faster
 - take up less memory

2/1/01

XML Parsing

12

Performance and Memory

- Therefore, when high performance and low-memory are the most important criteria, use a non-validating parser.
- Examples:
 - Java applets
 - Palm Pilot Applications

2/1/01

XML Parsing

13

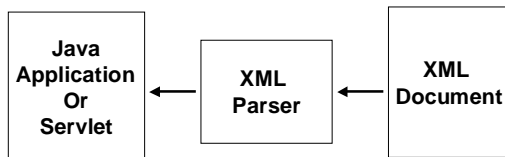
XML Interfaces

2/1/01

XML Parsing

14

General Architecture



The Parser sits in the middle of your application and your data. What's the best way to extract that data?

2/1/01

XML Parsing

15

XML Interfaces

- Broadly, there are two types of interfaces provided by XML Parsers:
 - Object/Tree Interface
 - Event Based Interface
- Let's examine each of these in detail...

2/1/01

XML Parsing

16

Object/Tree Interface

- **Definition:** Parser reads the XML document, and creates an in-memory "tree" of data.
- **For example:**
 - Given a sample XML document on the next slide, what kind of tree would be produced?

2/1/01

XML Parsing

17

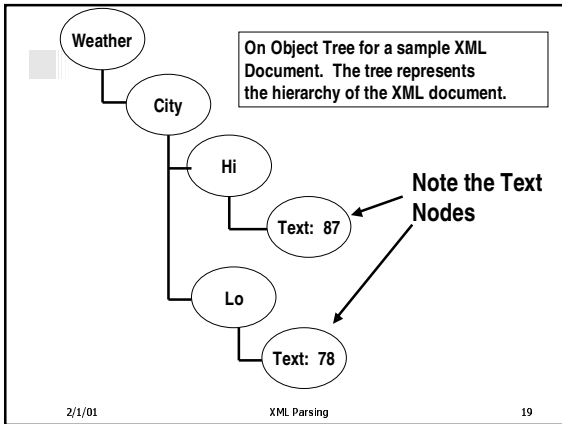
Sample XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE WEATHER SYSTEM "Weather.dtd">
<WEATHER>
  <CITY NAME="Hong Kong">
    <HI>87</HI>
    <LOW>78</LOW>
  </CITY>
</WEATHER>
```

2/1/01

XML Parsing

18



Event Based Parser

- **Definition:** Parser reads the XML document, and generates events for each parsing event.
- **For example:**
 - Given the same XML document, what kind of tree would be produced?

2/1/01 XML Parsing 20

Sample XML Document

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE WEATHER SYSTEM "Weather.dtd">
<WEATHER>
  <CITY NAME="Hong Kong">
    <HI>87</HI>
    <LOW>78</LOW>
  </CITY>
</WEATHER>
  
```

2/1/01 XML Parsing 21

XML Parsing Events

- **Events generated:**
 1. Start of <Weather> Element
 2. Start of <CITY> Element
 3. Start of <HI> Element
 4. Character Event: 87
 5. End of </HI> Element
 6. Start of <LOW> Element
 7. Character Event: 78
 8. End of </LOW> Element
 9. End of </CITY> Element
 10. End of </WEATHER> Element

2/1/01 XML Parsing 22

Event Based Interface

- For each of these events, the your application implements "event handlers."
- Each time an event occurs, a different event handler is called.
- Your application intercepts these events, and handles them in any way you want.

2/1/01 XML Parsing 23

Performance and Memory

- **Questions:**
 - Which parser is faster?
 - Which parser takes up less memory?
- **Tree based:**
 - more useful
 - slower
 - takes up more memory
- **Event based:**
 - less useful
 - faster
 - takes up much less memory

2/1/01 XML Parsing 24

■ Performance and Memory

- Therefore, when high performance and low-memory are the most important criteria, use an event-based parser.
- Examples:
 - Java applets
 - Palm Pilot Applications

2/1/01

XML Parsing

25

■ XML Interface Standards

2/1/01

XML Parsing

26

■ XML Interface Standards

- **Standards are important:**
 - Easier to create XML applications
 - You can swap parsers as your application evolves.
- There are two main XML Interface standards:
 - Tree Based: Document Object Model (DOM)
 - Event Based: Simple API for XML (SAX)

2/1/01

XML Parsing

27

■ DOM

- Document Object Model
- Tree Based Interface
- Developed by the W3C
- Supports both XML and HTML
- Originally specified using an IDL (Interface Definition Language).
 - Hence, DOM Versions exist for Java, JavaScript, C++, Perl, Python.

2/1/01

XML Parsing

28

■ SAX

- Simple API for XML
- Event Based
- Developed by volunteers on the xml-dev mailing list.
- <http://www.meggison.com/SAX/>

2/1/01

XML Parsing

29

■ Java XML Parsers

2/1/01

XML Parsing

30

Java XML Parsers

- There are currently dozens of XML Parsers written in Java.
- IBM XML Parser for Java
 - One of the first parsers. Widely used.
 - Validating and Non-validating Options
 - Supports DOM and SAX
 - <http://www.alphaworks.ibm.com/tech/xml4j>

2/1/01

XML Parsing

31

Java XML Parsers

- Sun Parser (Project X)
 - Not yet part of the Java standard. Need to download separately.
 - Validating and Non-validating Options
 - Supports DOM and SAX.
 - <http://java.sun.com/xml/>
- Aelfred XML Parser
 - Very lightweight, Low Memory Usage
 - Non-Validating, Event-Based Interface
 - Good for building Applets
 - <http://www.opentext.com/>

2/1/01

XML Parsing

32

Java XML Parsers

- Apache Xerces
 - Validating and Non-validating Options
 - Supports DOM and SAX.
 - <http://xml.apache.org>

2/1/01

XML Parsing

33

Java XML Parsers

- For a full list of XML Parsers, go to <http://www.xmlsoftware.com/parsers>
- Note that XML Parsers also exist for lots of other languages: C/C++, JavaScript, Python, Perl, etc.
- Most parsers support both DOM and SAX, and most have options for turning validation on or off.

2/1/01

XML Parsing

34

JAVA and XML Parsing

- Parsing
 - Enumerates the whole document, and
 - Validates the well-formedness
- Method 1: A Simple API for XML: SAX
 - Event driven interfaces must be filled in
 - startElement(...)
 - characters(...)
 - endElement(...)
- Can use parsers to do database selection

2/1/01

XML Parsing

35

SAX Event Trace

```
<?xml version="1.0">
<doc>
<para> Hello, world! </para>
</doc>
```

- start document
 - Start element: doc
- start element: para
 - Characters: Hello, world!
- end element: para
- end element: doc
- end document

2/1/01

XML Parsing

36

Using SAX: first, define an EventHandler.java program

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

public class EventHandler extends HandlerBase {
    boolean paraFlag;

    public void startElement (String tag, AttributeList attrs) throws SAXException {
        if (tag.equals("para")) {paraFlag=true;}
    }

    public void characters (char buf [], int offset, int len) throws SAXException {
        if (paraFlag) { //output to an html file, or whatever ...}
        }
        ...
    }
}
```

2/1/01

XML Parsing

37

SAX: SaxDriver.java (from Sun)

```
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.ParserFactory;
import com.sun.xml.parser.Resolver;

public class SaxDriver {
    // defines main etc...

    try {
        // turn filename into input source
        input = Resolver.createInputSource (new File (args[0]));

        // turn it into an in-memory object
        Parser myParser;
        myParser = ParserFactory.makeParser ();
        myParser.setDocumentHandler (new EventHandler ());
        myParser.parse (input);
    } catch ...
}
```

2/1/01

XML Parsing

38

Parsing with the IBM package

- Generating a parser and validating it against a dtd file referred to in an input xml file.

```
import com.ibm.xml.parser.Parser;
import // other stuff...

public class PrintDomTree {

    static public void main(String[] argv) {

        if (argv.length != 1) {
            // error checking stuff for argv
        }
        try {
            InputStream is = new FileInputStream(argv[0]);
            Parser parser = new Parser(argv[0]);
            if (parser.getNumberOfErrors () > 0) {
                system.out.println("Well Formed!");
            }
        }
        // other stuff...
    }
}
```

2/1/01

XML Parsing

39