XML Parsers

By Chongbing Liu

XML Parsers

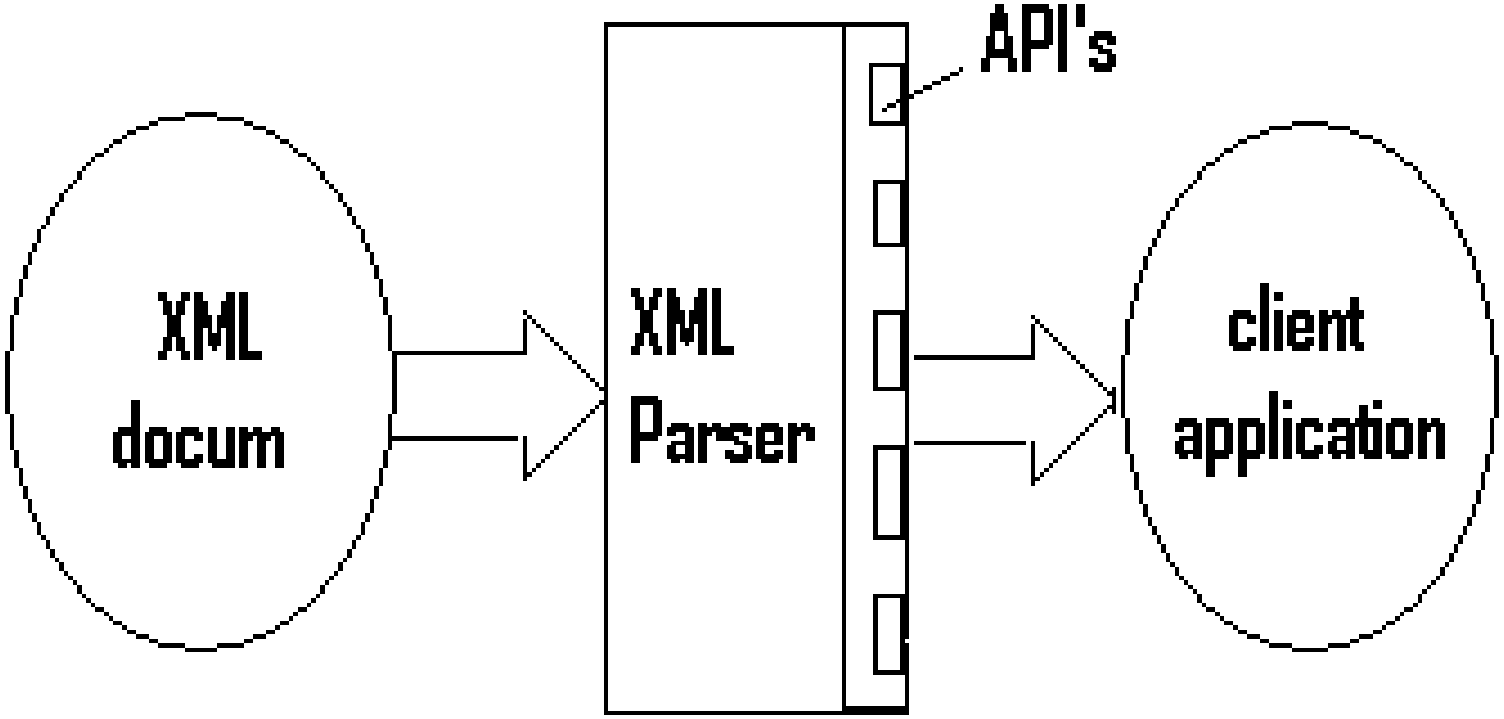
- What is a XML parser?
- DOM and SAX parser API
- Xerces-J parsers overview
- Work with XML parsers (example)

What is a XML Parser? 1

- ✓ It is a software library (or a package) that provides methods (or interfaces) for client applications to work with XML documents
- ✓ It checks the well-formattedness
- ✓ It may validate the documents
- ✓ It does a lot of other detailed things so that a client is shielded from that complexities

What is a XML Parser?

(*continued*)



DOM and SAX Parsers

in general

- ✓ DOM: *Document Object Model*
- ✓ SAX: *Simple API for XML*
- ✓ A DOM parser implements DOM API
- ✓ A SAX parser implement SAX API
- ✓ Most major parsers implement both DOM and SAX API's

DOM and SAX Parsers

DOM parsers

- DOM Document object
- Main features of DOM parsers

DOM and SAX Parsers

DOM Document Object

- A ***DOM document*** is an object containing all the information of an XML document

- It is composed of a tree (DOM tree) of nodes , and various nodes that are somehow associated with other nodes in the tree but are not themselves part of the DOM tree

DOM and SAX Parsers

DOM Document Object

- There are 12 types of nodes in a DOM *Document* object
  - *Document node*
  - *Element node*
  - *Text node*
  - *Attribute node*
  - *Processing instruction node*
  - …….

DOM and SAX Parsers

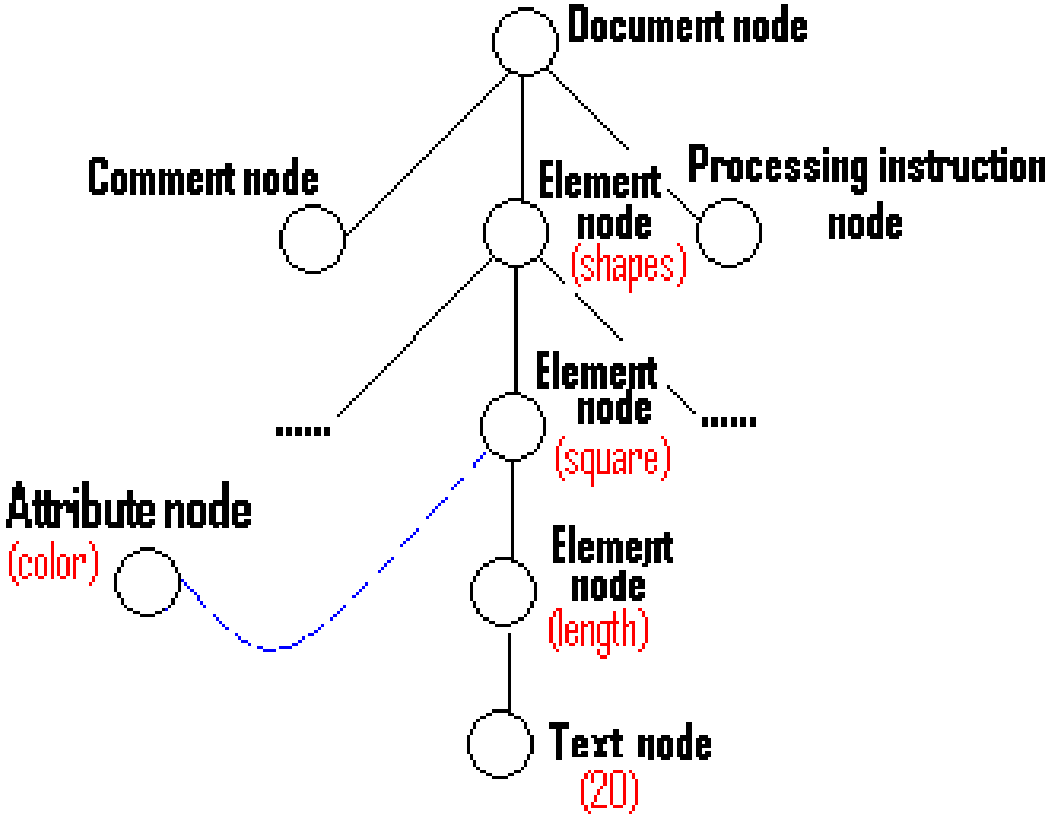
DOM parsers – continued (Appendix)

Sample XML document

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="test.css"?>
<!-- It's an xml-stylesheet processing instruction. -->
<!DOCTYPE shapes SYSTEM "shapes.dtd">
<shapes>
        ……
        <squre color="BLUE">
            <length> 20 </length>
        </squre>
        ……
 </shapes>
```

DOM and SAX Parsers

DOM parsers – continued (Appendix)



DOM and SAX Parsers

main features of DOM parsers

- ✓ A DOM parser creates an internal structure in memory which is a ***DOM document*** object
- ✓ Client applications get the information of the original XML document by invoking methods on this ***Document*** object or on other objects it contains
- ✓ DOM parser is tree-based (or DOM obj-based)
- ✓ Client application seems to be pulling the data actively, from the data flow point of view

DOM and SAX Parsers

main features of DOM parsers (cont.)

✓ Advantage:

(1) It is good when random access to widely separated parts of a document is required

(2) It supports both read and write operations

✓ Disadvantage:

(1) It is memory inefficient

(2) It seems complicated, although not really

DOM and SAX Parsers

SAX parsers

- ✓ It does not first create any internal structure
- ✓ Client does not specify what methods to call
- ✓ Client just overrides the methods of the API and place his own code inside there
- ✓ When the parser encounters start-tag, end-tag,etc., it thinks of them as **events**

DOM and SAX Parsers

SAX parsers (cont.)

- ✓ When such an **event** occurs, the handler automatically ***calls back*** to a particular method overridden by the client, and feeds as arguments the method what it sees
- ✓ SAX parser is event-based,it works like an event handler in Java (e.g. MouseAdapter)
- ✓ Client application seems to be just receiving the data inactively, from the data flow point of view

DOM and SAX Parsers

SAX parsers (cont.)

- ✓ Advantage:
  - (1) It is simple
  - (2) It is memory efficient
  - (3) It works well in stream application
- ✓ Disadvantage:

The data is broken into pieces and clients never have all the information as a whole unless they create their own data structure

Appendix: Call back in Java

```
class MyMouseListener extends java.awt.event.MouseAdapter {
      /** Overriding the method mousePressed(). */
      public void mousePressed(java.awt.event.MouseEvent event) {
        .....do something here after the mouse is pressed …….
      }

      /** Overriding the method mousePressed(). */
      public void   mouseReleased(java.awt.event.MouseEvent event) {
         .....do something here after the mouse is released …….
      }
}
MyMouseListener Listener = new MyMouseListener();
java.awt.Button MyButton=new java.awt.Button("ok");
MyButton.addMouseListener(Listener);
```

DOM and SAX Parsers

Xerces-J Parser Overview 1

- ✓ It is a Java package
- ✓ Provides two parsers, one is a **DOM parser** and another is a **SAX parser**
- ✓ It is a validating parser
- ✓ It fully supports DOM2 and SAX2, and partially supports DOM3 (W3C XML Schema)
- ✓ It is very popular

Xerces-J Parser Overview

package structure

java.lang.Object

| +--org.apache.xerces.framework.XMLParser

| +-- org.apache.xerces.parsers.DOMParser

+-- org.apache.xerces.parsers.SAXParser

Xerces-J Parser Overview

DOMParser methods

- Void parse (java.lang.String systemId)

  Parses the input source specified by the given system identifier.

- Document getDocument()

  Returns the document

- ……

Xerces-J DOMParser

DOM interfaces

- **Document**
- **Element**
- **Attr**
- **NodeList**
- **ProcessingInstruction**
- **NamedNodeMap**
- **. . . . .**

Xerces-J Parser Overview

SAXParser methods

- Void parse (java.lang.String systemId)

  Parses the input source specified by the given system identifier.

- Void setContentHandler(Contenthandler handler)

  Allow an application to register a content event handler.

- Void setErrorHandler(Errorhandler handler)

  Set error handler.

- ……

Xerces-J Parser Overview

SAXParser interfaces

- ContentHandler
- DTDHandler
- EntityResolver
- ErrorHandler

Work with XML Parsers

Example

Task: Extract all information about circles

```
<?xml version="1.0"?>
<!DOCTYPE shapes SYSTEM “shapes.dtd">
<shapes>
        <circle color=“BLUE”>
            <x> 20 </x>
            <y> 20 </y>
            <radius> 20 </radius>
        </circle>
</shapes>
```

Example

DOMParser:create client class

```
public class shapes_DOM {
    static int numberOfCircles = 0;
    static int x[] = new int[1000];
    static int y[] = new int[1000];
    static int r[] = new int[1000];
    static String color[] = new String[1000];

    public static void main(String[] args) {
        ……
    }
}
```

Example

(DOMParser: *create a DOMParser)* 2

```
import org.w3c.dom.*;
import org.apache.xerces.parsers.DOMParser;
 public class shapes_DOM {
      ……
     public static void main(String [ ] args ) {
           try{
                DOMParser parser=new DOMParser();
                parser.parse(args[0]);
                Document doc=parser.getDocument();
                ……
           }    catch (Exception e) {
                     e.printStackTrace(System.err);
                }
      }
}
```

Example

(DOMParser: *get all the circle nodes)* 3

```
NodeList nodelist =
  doc.getElementsByTagName("circle");

numberOfCircles = nodelist.getLength();
```

Example

(DOMParser: *iterate over circle nodes)* 4

```
for(int i=0; i<nodelist.getLength(); i++) {

        Node node = nodelist.item(i);
              .
              .
              .
}
```

Example

(DOMParser: *get color attribute)* 5

```
25    NamedNodeMap attrs = node.getAttributes();
26    if(attrs.getLength()!=0)
26'       color[i]=
      (String)attrs.getNamedItem("color").getNodeValue();
```

Example

(DOMParser: *get child nodes)* 6

```
27 // get the child nodes of a circle
28 NodeList childnodelist = node.getChildNodes();

29 // get the x and y
30 for(int j=0; j<childnodelist.getLength(); j++) {
31   Node childnode = childnodelist.item(j);
32   Node textnode = childnode.getFirstChild();
33   String childnodename = childnode.getNodeName();
34   if(childnodename.equals("x"))
35     x[i]=Integer.parseInt(textnode.getNodeValue().trim());
36   else if(childnodename.equals("y"))
37     y[i]=Integer.parseInt(textnode.getNodeValue().trim());
38   else if(childnodename.equals("radius"))
39     r[i]=Integer.parseInt(texxtnode.getNodeValue().trim())
40 }
```

Example

(SAXarser: create client class) 1

```
public class shapes_SAX extends DefaultHandler {
    static int numberOfCircles = 0;
    static int x[] = new int[1000];
    static int y[] = new int[1000];
    static int r[] = new int[1000];
    static String color[] = new String[1000];

    public static void main(String[] args) {
        ……
    }
}
```

Example

(SAXParser: *create a SAXParser)* 2

```
import org.xml.sax.*;
import org.xml.sax.helpers.DefaultHandler;
import org.apache.xerces.parsers.SAXParser;
 public class shapes_SAX extends DefaultHandler{
    public static void main(String [ ] args ) {
       try{
          shapes_SAX SAXHandler = new shapes_SAX();
          SAXParser parser = new SAXParser();
          parser.setContentHandler(SAXHandler);
          parser.parse(args[0]);
       }    catch (Exception e) { … … }
    }
}
```

Example

(SAXParser: *override methods of interest)* 3

- `startDocument()` `endDocument()`
- `startElement()` `endElement()`
- `startCDATA()` `endCDATA()`
- `startDTD()` `endDTD()`
- `characters()`
- … …

Example

(SAXParser: *override startElement()*) 4

```
21 public void startElement(String uri, String localName,
                            String rawName, Attributes attributes) {
22   if(rawName.equals("circle" )
23      color[numberOfCircles]=attributes.getValue("color");
26   else if(rawName.equals("x"))
27     flagX = 1;
28   else if(rawName.equals("y"))
29     flagY = 1;
30   else if(rawName.equals("radius"))
31     flagR = 1;
32 }
```

Example

(SAXParser: *override endElement())*

```
33 public void endElement(String uri, String
                  localName, String rawName) {
34            numberOfCircles +=  1;
35 }
```

Example

(SAXParser: *override characters()*) 5

```
36 public void characters(char characters[], int start,
                          int length) {
38          String characterData =
39              (new String(characters,start,length)).trim();
42          if(flagX==1) {
43              x[numberOfCircles] = Integer.parseInt(characterData);
                flagX=0;
            }
44          if(flagY==1) {
45              y[numberOfCircles] = Integer.parseInt(characterData);
                flagY=0;
            }
46          if(flagR==1) {
47              r[numberOfCircles] = Integer.parseInt(characterData);
                flagR=0;
            }
49 }
```

Example

(SAXParser: *override endDocument()*)

```
50 public void endDocument() {
51          // print the result
52     System.out.println("circles="+numberOfCircles);
53     for(int i=0;i<numberOfCircles;i++) {
54        String line="";
55       line=line+"(x="+x[i]+",y="+y[i]+",r="+r[i]
                  +",color="+color[i]+")";
56        System.out.println(line);
57     }
58 }
```

DOM and SAX Parsers