

We'll see how to parse a generic XML file using kXML library.

Source code: XmlNode class

First, we'll define an XmlNode class to represent a generic XmlNode. We'll define 2 node types:

- Text nodes: nodes without a name, containing only a textual value
- Element nodes: nodes with a tag name, containing children nodes

Each node will have also a list of attributes.

```
public class XmlNode
{
    public static final int TEXT_NODE = 0;
    public static final int ELEMENT_NODE = 1;

    public int nodeType = 0;
    public String nodeName = null;
    public String nodeValue = null;
    public Vector children = null;
    public Hashtable attributes = null;

    public XmlNode(int nodeType)
    {
        this.nodeType = nodeType;
        this.children = new Vector();
        this.attributes = new Hashtable();
    }
    public String[] getAttributeNames()
    {
        String[] names = new String[attributes.size()];
        Enumeration e = attributes.keys();
        int i = 0;
        while(e.hasMoreElements())
        {
            [i] = (String)e.nextElement();
            ++i;
        }
        return names;
    }
    public void setAttribute(String key, String value)
    {
        put(key, value);
    }
    public String getAttribute(String key)
    {
        return (String)attributes.get(key);
    }

    public void addChild(XmlNode child)
    {
        this.children.addElement(child);
    }
}
```

Source code: GenericXmlParser class

GenericXmlParser is the class with which we'll be able to parse XML files. It has only one public method, parseXML(), that will accept as arguments:

- a KXmlParser instance to do parsing
- a boolean that defines if whitespaces-only children must be ignored

```
import org.kxml2.io.KXmlParser;
import org.xmlpull.v1.XmlPullParser;

public class GenericXmlParser
{
    public XmlNode parseXML(KXmlParser parser,
        boolean ignoreWhitespaces) throws Exception
    {
        next(); parser.

return _parse(parser, ignoreWhitespaces);
}

    XmlNode(KXmlParser parser,
        boolean ignoreWhitespaces) throws Exception
    {
        = new XmlNode(XmlNode.ELEMENT_NODE);

if(parser.getEventType() != XmlPullParser.START_TAG)
{
throw new Exception("Illegal XML state: " +
                    parser.getName() + ", " + parser.getEventType());
}
else
{
    nodeName = parser.getName();

for(int i = 0; i < parser.getAttributeCount(); i++)
{
    setAttribute(parser.getAttributeName(i), parser.getAttributeValue(i));
}

        next(); parser.

while(parser.getEventType() != XmlPullParser.END_TAG)
{
if(parser.getEventType() == XmlPullParser.START_TAG)
{
    addChild(_parse(parser, ignoreWhitespaces));
}
else if(parser.getEventType() == XmlPullParser.TEXT)
{
if(!ignoreWhitespaces || !parser.isWhitespace())
{
            = new XmlNode(XmlNode.TEXT_NODE); XmlNode child

            nodeValue = parser.getText(); child.

            addChild(child); node.
}
}

        next(); parser.
}
}
```

```
}  
return node;  
}  
}
```

Source code: sample usage

To use the classes defined above, you can simply do as follows:

```
InputStreamReader reader = new InputStreamReader(getClass().getResourceAsStream("/test3.xml"));  
  
KXmlParser parser = new KXmlParser();  
  
parser.setInput(reader);  
  
GenericXmlParser gParser = new GenericXmlParser();  
  
XmlNode xml = gParser.parseXML(parser, true);
```

And, if you want to dump out the parsed XML data, you can use the following method:

```
void dumpXML(XmlNode node, int deep)  
{  
for(int i = 0; i < deep; i++)  
{  
System.out.print(" ");  
}  
System.out.print(node.nodeName + " - ");  
  
if(node.nodeValue != null)  
{  
System.out.print("(" + node.nodeValue + ") - ");  
}  
String[] attributes = node.getAttributeNames();  
  
for(int i = 0; i < attributes.length; i++)  
{  
System.out.print(attributes[i] + ": " + node.getAttribute(attributes[i]) + ", ");  
}  
  
System.out.println();  
  
for(int i = 0; i < node.children.size(); i++)  
{  
    ((XmlNode)node.children.elementAt(i), deep + 1);  
}  
}
```