Parsing XML from Java

Short introduction

What is “parsing”?

A “parser” is a component which takes some input and turns it into some datastructure like an object or a tree etc. It allows us to check for validity and gives us a structured representation of the input.

So, take some text, like XML, build a tree from it, and check that it is well-formed (syntactically correct).

When do we need to “parse” XML?

If our Java application gets data from some service or file in the form of an XML document, we need to transform it from XML to some kind of representation in Java objects.

The goal is to get the XML from somewhere and create a representation of that data in Java.

So we need to “parse” the XML and traverse the tree and create some Java data type object from it (like a List for instance).

Example with an XML document of users

Let’s say our application needs a list of users for some system. A user has a name, an email address and a username. The application gets the data in the form of an XML document.

The XML with the users

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<users>
  <user>
    <name>Kalle Anka</name>
    <email>donald@email.dt</email>
    <username>donaldd</username>
  </user>
  <user>
    <name>Joakim von Anka</name>
    <email>scrooge@email.dt</email>
    <username>onkelscrooge</username>
  </user>
  <user>
    <name>Arne Anka</name>
    <email>arne@email.com</email>
    <username>arneanka</username>
  </user>
</users>
```

Parsing the xml document in Java

Our parser must read in the XML text and create a tree representation of it, traverse the tree and do something with every node.

This is done in steps:

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
// We need to parse the bytes of the XML (A String reference called raw)
Document document = builder.parse((new ByteArrayInputStream(raw.getBytes()));
NodeList nodeList = document.getDocumentElement().getChildNodes();
```

Explaining the first steps

```
// We need a factory to get a document builder
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();

// We need to parse the bytes of the XML (A String reference called raw)
Document document = builder.parse(new ByteArrayInputStream(raw.getBytes()));

// Get the root element, and from that, get all the child nodes (<user> elements)
NodeList nodeList = document.getDocumentElement().getChildNodes();
```

Looping through the nodes

```
for (int i = 0; i < nodeList.getLength(); i++) {
  Node node = nodeList.item(i);
  if (node.getNodeType() == Node.ELEMENT_NODE) { // must be an element!
    Element elem = (Element) node;
    String name = elem.getElementsByTagName("name")
      .item(0).getChildNodes().item(0).getNodeValue();
    String email = elem.getElementsByTagName("email")
      .item(0).getChildNodes().item(0).getNodeValue();
    String userName = elem.getElementsByTagName("username")
      .item(0).getChildNodes().item(0).getNodeValue();
    // Do something with the strings...
}
```

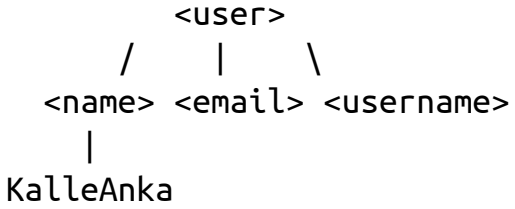
Deeper look at getting the text of a node

```
Element elem = (Element) node;
String name = elem.getElementsByTagName("name")
  .item(0).getChildNodes().item(0).getNodeValue();
```

```
<user> ← node
  <name>Kalle Anka</name> ← “<name>” is elements[0]
                            the text is the first child of
                                        elements[0]
  <email>donald@email.dt</email>
  <username>donaldd</username>
</user>
```

The tree representation

```
<user>
  <name>Kalle Anka</name>  ← we want to get to the value Kalle Anka
  <email>donald@email.dt</email>
  <username>donaldd</username>
</user>
```



So, this user element has a *list* of one element with name “name”. That element has a *list* of one child nodes (of type text) with the value Kalle Anka.

It *could* have more name elements (since it’s XML), that’s why everything is a list!

Deeper look at getting the text of a node

```
Element elem = (Element) node;
String name = elem.getElementsByTagName("name")
  .item(0).getChildNodes().item(0).getNodeValue();
```

getElementsByTagName("name") returns a list with only one element:
[<name>Kalle Anka</name>]
getElementsByTagName("name").item(0) is that only element
Now, to get the text part, we need to get the children of this element and use the first one:
getElementsByTagName("name").item(0).getChildNodes().item(0)
And get the value:
getElementsByTagName("name").item(0).getChildNodes().item(0)
.getNodeValue() ← Kalle Anka

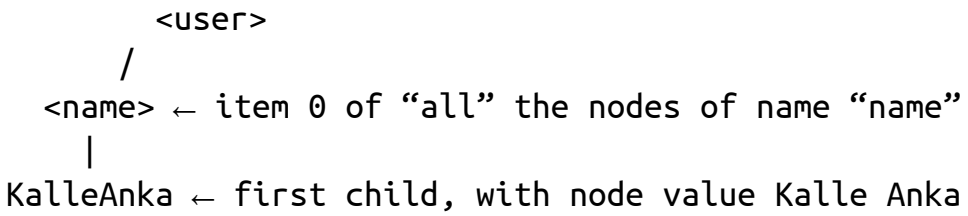
Isn’t there a shorter way?

```
Element elem = (Element) node;
String name =
elem.getElementsByTagName("name").item(0).getChildNodes().item(0)
    .getNodeValue();
```

We could shorten this to:

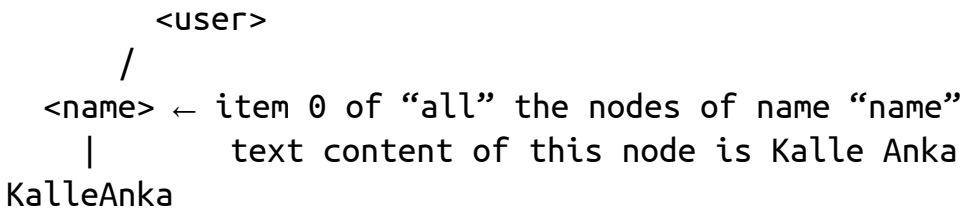
```
elem.getElementsByTagName("name").item(0).getFirstChild().getNodeValue()
Or even:
elem.getElementsByTagName("name").item(0).getTextContent()
```

Look at the tree again




```
elem.getElementsByTagName("name").item(0).getFirstChild().getNodeValue()
```

Look at the tree again




```
elem.getElementsByTagName("name").item(0).getTextContent()
```

Ok, ok, we got it. Show us the code!

```
try {
  DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
  DocumentBuilder builder = factory.newDocumentBuilder();
  Document document = builder.parse("users.xml"); // file in current directory
  NodeList nodeList = document.getDocumentElement().getChildNodes();
  for (int i = 0; i < nodeList.getLength(); i++) {
    Node node = nodeList.item(i);
    if (node.getNodeType() == Node.ELEMENT_NODE) {
      Element elem = (Element) node;
      String name = elem.getElementsByTagName("name").item(0).getTextContent();
      String email = elem.getElementsByTagName("email").item(0).getTextContent();
      String userName = elem.getElementsByTagName("username").item(0).getTextContent();
      System.out.println(name + " " + email + " " + userName);
    }
  }
}catch(SAXException|IOException|ParserConfigurationException e){
  System.err.println("Error parsing XML: " + e.getMessage());
}
```

Import statements

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
// factory.newDocumentBuilder() can throw a ParserConfigurationException
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException; // builder.parse()
import java.io.IOException;      // builder.parse()
```

Example run

```
$ cat users.xml && java client.SmallParser
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<users>
  <user>
    <name>Kalle Anka</name>
    <email>donald@email.dt</email>
    <username>donaldd</username>
  </user>
  <user>
    <name>Joakim von Anka</name>
    <email>scrooge@email.dt</email>
    <username>onkelscrooge</username>
  </user>
  <user>
    <name>Arne Anka</name>
    <email>arne@email.com</email>
    <username>arneanka</username>
  </user>
</users>
name: Kalle Anka email: donald@email.dt userName: donaldd
name: Joakim von Anka email: scrooge@email.dt userName: onkelscrooge
name: Arne Anka email: arne@email.com userName: arneanka
```

Get the source code

https://github.com/progund/java-web/tree/master/java-xml/xml-intro/presentation-sources

You can download the files one-by-one by clicking on the links for the files and then clicking on “raw”. Or clone the entire java-web repository (lots of files).

https://raw.githubusercontent.com/progund/java-web/master/java-xml/xml-intro/presentation-sources/users.xml

https://raw.githubusercontent.com/progund/java-web/master/java-xml/xml-intro/presentation-sources/client/SmallParser.java