XML Parsing Basics

Basics and understanding

Malintha Adikari

Software Engineer

What is XML Parsing

An XML parser is the piece of software that reads XML files and makes the information from those files available to applications and programming languages.

XML parsing approaches...

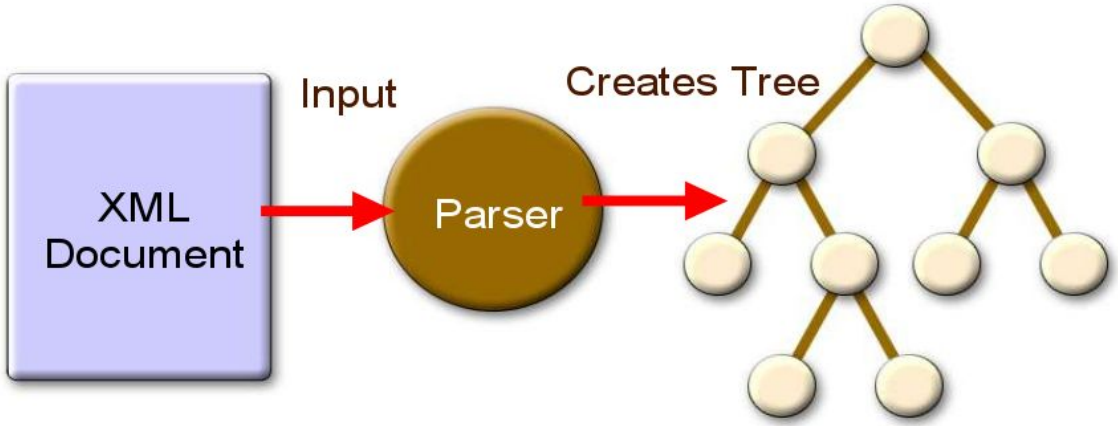
- **Tree-based APIs**
  - Object (DOM, JDOM….etc)
- **Event based APIs**
  - PUSH (SAX)
  - PULL(Stax)

XML parsing approaches...

- **Tree-based APIs**
  - The whole XML document is parsed and a model of that is built in memory
  - made it possible to go back and forward through an XML which is already read
  - The model they build is usually larger than the original XML document, thus duplicating and wasting memory

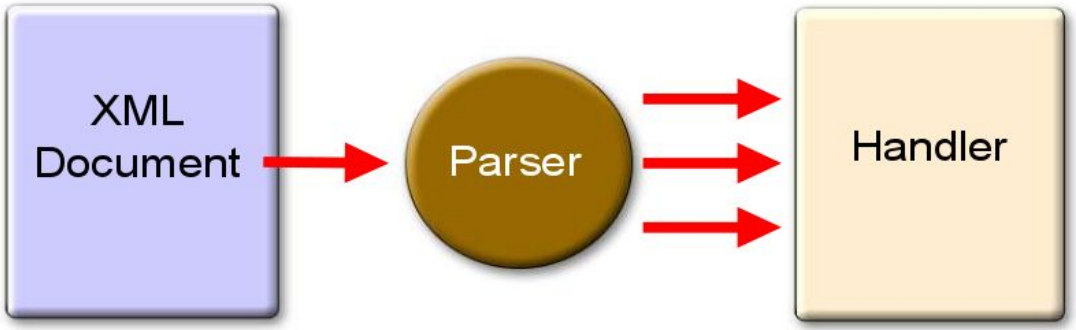
- **Event based APIs**
  - event based parser parses the whole XML document and throws "events" depending on the information content of the XML

Object model (DOM Tree)



The Document Object Model parser is a hierarchy-based parser that creates an object model of the entire XML document, then hands that model to you to work with

Push model (SAX)

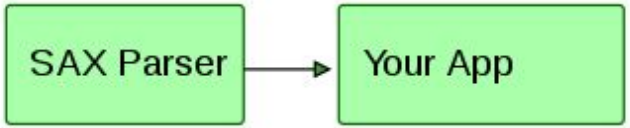


An event-based sequential access parser API that only operates on portions of the XML document at any one time

Push model (SAX)

- SAX is a push style API
- This means that the SAX parser iterates through the XML and calls methods on the handler object provided by you. For instance, when the SAX parser encounters the beginning of an XML element, it calls the startElement on your handler object
- It "pushes" the information from the XML into your object. Hence the name "push" style API. This is also referred to as an "event driven" API
- Your handler object is notified with event-calls when something interesting is found in the XML document ("interesting" = elements, texts, comments etc.).

The SAX parser push style parsing is illustrated here:



Push model (SAX)

- the parser continuously pushes events to the calling application until it finishes reading the whole XML document
- This is more efficient than DOM in terms of memory
- once started, it goes to the end of the document and the caller must be ready to handle all the events in one shot
- The caller that invokes the parser has no control over the parsing process.
- Once started, tree based or event based push models consume the whole data stream at once.

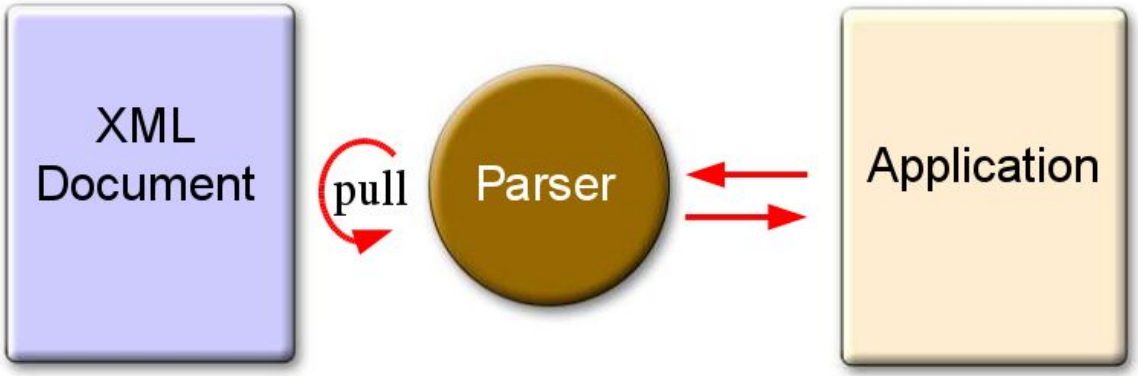
Push model (SAX)

- It works by iterating over the XML and call certain methods on a "listener" object when it meets certain structural elements of the XML

- For instance, it will call the listener object for the following "events":

  - startDocument
  - startElement
  - characters
  - comments
  - processing instructions
  - endElement
  - endDocument

Pull model

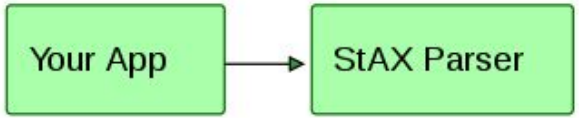


The Java Architecture for XML Binding maps Java classes to XML documents and allows you to operate on the XML in a more natural way

Pull model (StaX parser)

- StAX is a pull style API.
- This means that you have to move the StAX parser from item to item in the XML file yourself, just like you do with a standard Iterator or JDBC ResultSet.
- You can then access the XML information via the StAX parser for each such "item" encountered in the XML file ("item" = elements, texts, comments etc.).
- unlike in SAX, the client has the full control to start, proceed, pause, and resume the parsing process

The StAX parser pull style parsing is illustrated here:



String operations*

String operations on a loaded XML document to manually find bits of information within the XML as a String; for instance, using the String class's indexOf and other built-in methods. This is not a scalable or reusable solution

Let’s start with DOM parser….

Document Object Model (DOM)

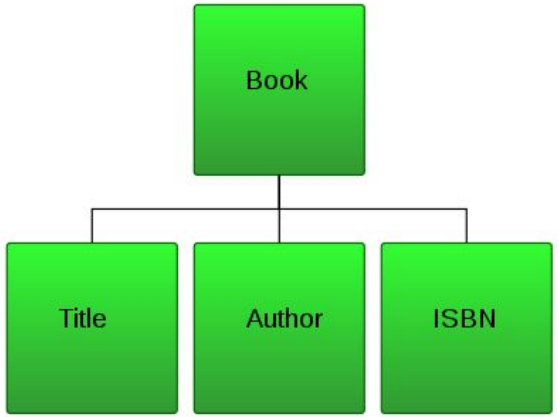
- The Java DOM API for XML parsing is intended for working with XML as an object graph in memory - a "Document Object Model (DOM)"
- The parser traverses the XML file and creates the corresponding DOM objects
- These DOM objects are linked together in a tree structure
- Then you can traverse the DOM structure back and forth as you see fit

DOM structure

Here is an example XML file, and a DOM tree that illustrates the principle of turning XML into DOM:

```
<book>
    <title>Fun Software</title>
    <author>Jakob Jenkov</author>
    <ISBN>0123456789</ISBN>
</book>
```

And the corresponding DOM structure:



DOM - 3 pieces of XML

1. Elements (sometimes called tags)
2. Attributes
3. The data (also called values) that the elements and attributes describe

Start with DOM parser

1. Creating a Java DOM XML parser

Creating a Java DOM XML parser is done using the javax.xml.parsers.DocumentBuilderFactory class. Here is an example:

```
DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = null;
try {
   builder = builderFactory.newDocumentBuilder();
} catch (ParserConfigurationException e) {
   e.printStackTrace();
}
```

Start with DOM parser

2. Parsing an XML file into a DOM tree

Parsing an XML file into a DOM tree using the DocumentBuilder is done like this:

```
try {
   Document document = builder.parse( new FileInputStream("/path/to/your/file.xml"));
} catch (SAXException e) {
   e.printStackTrace();
} catch (IOException e) {
   e.printStackTrace();
}
```

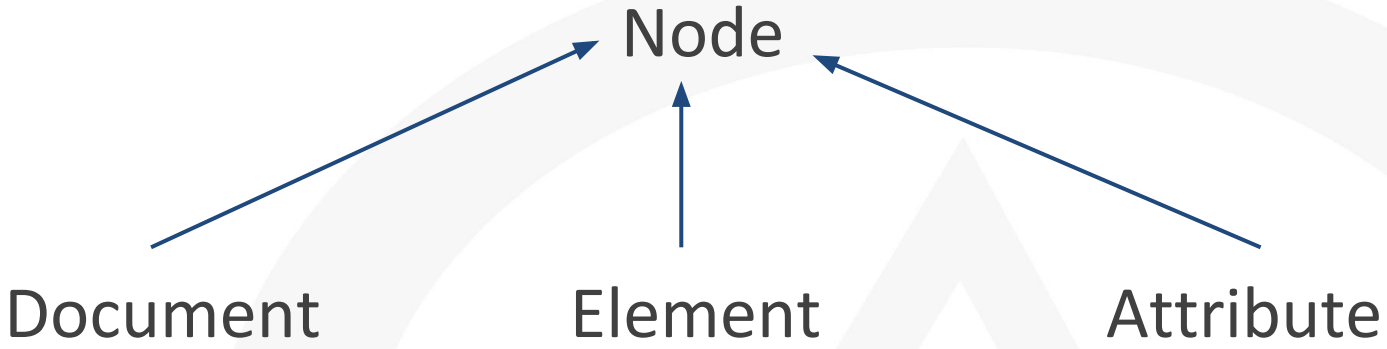
You are now ready to traverse the Document instance you have received from the DocumentBuilder

DOM document element

- A DOM object contains a lot of different nodes connected in a tree-like structure.
- At the top is the Document object.
- The Document object has a single root element, which is returned by calling **getDocumentElement()**

```
Element rootElement = document.getDocumentElement();
```

Node,Element,Attribute .....



Demo

When should I use DOM

- When you need random access to document data
  - If random access to information is crucial, it is better to use theDOM to create a tree structure for the data in memory.
- When you want to implement complex searches
  - maintain data structures holding context information such as attributes of current element
- No SAX implementation in current browsers
  - Microsoft's Internet Explorer
- When you need to perform XSLT transformations
- When you want to modify and save XML
- DOM allows you to create or modify a document in memory, as well as read a document from an XML source file

When should I use SAX

- **When your documents are large**:Perhaps the biggest advantage of SAX is that it requires significantly less memory to process an XMLdocument than the DOM. With SAX, memory consumption does not increase with the size of the file

- **When you need to abort parsing**:Because SAX allows you to abort processing at any time, you can use it to create applications that fetch particular data

- **When you want to retrieve small amounts of information**:For many XML-based solutions, it is not necessary to read the entire document to achieve the desired results. Scanning only a small percentage of the document results in a significant savings of system resources.

- **When you want to create a new document structure**:In some cases, you might want to use SAX to create a data structure using only high-level objects, such as stock symbols and news, and then combine the data from this XML file with other news sources.Rather than build a DOM structure with low-level elements,attributes, and processing instructions, you can build the document structure more efficiently and quickly using SAX

When should I use StAX

Most of the applications that process XML benefit from stream parsing, and most of the time does not require the entire DOM model in the memory. Having mentioned that as the main advantage we have in pull parsing, let's look at the other aspects.

- the client gains control of this parsing model and the parsing happens according to client requirements. However in the pull model, the client is "pushed" with data, irrespective of whether it is needed.

- Pull parsing libraries are much smaller compared to the respective push libraries, and even the client code that interacts with these libraries are small, even for complex documents.

- Filtering of elements is easier, as the client knows that when a particular element comes in, he has time to make the decisions

Questions?

Exercise

1. Download “automation.xml” file from following svn location
   https://svn.wso2.org/repos/wso2/carbon/platform/branches/turing/platform-integration/test-automation-framework-2/org.wso2.carbon.automation.engine/4.3.0/src/main/resources/automation.xml

2. Load data in the xml file to your data structure using DOM parser. Develop your own API to retrieve nodeValues

   ex: getNodeValue(String value, Node parent)

* Provided xml file has comments. Identify the effects of comment over building the DOM tree. Remove xml comments before you build the DOM tree

Get a XML and remove comments before you build it to DOM. What Happens ? Try to fix it

North America

Cisco, ATC, Acquity Group, American Family Insurance, Children's Hospital Boston, SAIC, Institut national de santé publique Québec, MEMC, Trimble, UMUC, Concur, JPMorganChase, Prudential, XO, VERIO, Blue Cross of Idaho, BNY Mellon, dg, AAA, Front Porch Digital, imshealth, Intermountain Healthcare, StubHub!, Kaiser Permanente, Boeing, Expedia, Tractor Supply Co, UBS, Noel Group, ebay, Flexera Software, Regions, neustar, barclaycard, NATO OTAN

Europe

BBVA, Brand Implementation Group, Cineca, Iberia, Berlin Airports, Volvo, suva, Truvo, csi, digipolis, Soft Design, GAD, A.M.A., AllianceBernstein, Tag, Mercedes-Benz, Securitas, T-Systems, Salzburg AG, O2, orange, British Airways, RosEvroBank, Alfa-Bank, eurotaxglass's

Middle East and Asia-Pacific

Commercial Credit, Sri Lanka Telecom Mobitel, Cognizant, ICTA ideas actioned, ODEL, Dialog, Dubai Smart Gov, nlc, onesteel, Australian Government Department of Health and Ageing, JESAL, VIB

South America

SHF, BI&P Banco Industrial & Partners

Contact us !