Daniela Martignani
Peter Miele
Jennifer Shaska

CSE551 HW1

Question 1: Extend given sample .xml file by adding sold date of item (month, day, and year). Explain the importance of XML?

See attached sample.xml file.

Importance of XML:
XML is important as it provides a standard format to communicate the format of data. With XML a document can define its own structure, or rely on a pre-defined structure in another document. By doing this, clients of the data originator are able to understand the structure of the data and thus more easily process and correctly interpret the data for their own use. With the extensible aspect of XML, a document producer can add to a document and previous users of the document will not be affected. The main importance of XML is that it provides a means for two parties to agree on a format for data that will be communicated, so that each can read and respond correctly.

Question 2: Compare and contrast StAX parser with SAX parser. Write a 3 page summary on integrated approach of JSON and XML parsing named StAXON - JSON via StAX.

Comparing StAX with other JAXP API's:

StAX can be compared to other standard JAXP API's by the way it works, the principles it follows and the features it supports. StAX has benefits that involve both main parsers (DOM and SAX), but it can be more compared/contrasted with SAX, by their similar nature. StAX compares to SAX in that it is not as powerful as DOM or TrAX, but it doesn't require a lot of memory or processor load. The efficient use of resource is what makes StAX similar to SAX and more suitable for applications that have a limited amount of physical resources.

By comparing StAX with SAX, we can say that StAX-enabled clients are easier to code than SAX-clients and StAX parser code is smaller than SAX [1]. Meaning, once SAX starts reading an XML file, it will move through the entire document, calling the handler for each XML event it encounters. When StAX reads an XML file, the handler dictates when StAX should move on to the next element, and if the handler stops making calls, then StAX will not go through the entire XML file. This allows StAX to do sub-parsing more easily and in a cleaner fashion than SAX.

SAX stands for 'Simple API for XML,' and many people assume that StAX stands for 'Streaming API for XML,' but they both parse XML files in a streaming fashion, in contrary to DOM, which is a model that involves creating in-memory objects representing an entire document tree and the complete infoset state for an XML document [1].

One main difference between StAX and SAX is that SAX can only be used for reading XML, while StAX can be used for both reading and writing XML. StAX is a bidirectional API (read and

write XML), while SAX is only able to read XML documents [1]. This allows StAX to be more flexible and comparable to other APIs like DOM parser. It is also a good improvement over SAX and a feature that makes StAX be more powerful and possibly more useful than SAX in certain scenarios.

Another main difference is that StAX follows the principle of a pull API, while SAX is basically a push API [1]. The SAX parser pushes the XML data to the handler, whereas with the StAX parser, the handler pulls the XML data elements on demand. SAX is considered to be push-style because the SAX parser automatically goes through the whole XML document and you catch the events it fires with your handler. For example, it triggers the startDocument, startElement, and endDocument routines itself and your handler just listens for them. StAX is considered to be pull-style because you have to specifically tell it to move the cursor from one element to the next.

One advantage that SAX has over StAX is that SAX supports schema validation, although it needs to be explicitly 'turned on' by creating a schema object and attaching an instance of the schema to the SAXParserFactory. But the main disadvantage of SAX compared to StAX is that StAX can write XML files while SAX only has the ability to read them.

To summarize, the following table [1] contains the main JAXP APIs and the differences between them in terms of features.

XML Parser API Feature Summary

Feature	StAX	SAX	DOM	TrAX
API Type	Pull, streaming	Push, streaming	In memory tree	XSLT Rule
Ease of Use	High	Medium	High	Medium
XPath Capability	No	No	Yes	Yes
CPU and Memory Efficiency	Good	Good	Varies	Varies
Forward Only	Yes	Yes	No	No
Read XML	Yes	Yes	Yes	Yes
Write XML	Yes	No	Yes	Yes
Create, Read, Update, Delete	No	No	Yes	No

As a conclusion, StAX offers good features of both DOM and SAX, and in some cases, it can be preferable over either DOM or SAX, especially when dealing with JSON files.

StAXON

StAXON provides an implementation of the StAX API that can switch between JSON and XML as the inputs and outputs. It does this by adapting the differences of XML into JSON constructs, such as attributes in XML become fields prefixed with '@' with the corresponding value becoming the '$' property of the JSON object[2].

As JSON and XML follow a very similar structure this means that the aspects of SAX that work with XML can work with JSON. In JSON and XML each item being represented has a start and an end. In XML this is generally done with opening and closing tags that wrap an element, in JSON the usage of braces denote the start and stop of an element. In this way the SAX methods of startElement and endElement can be applied to both XML and JSON.

StAXON at its core is simply a set of rules that govern the translation between JSON and XML. However, XML does support additional features such as namespaces that JSON does not have in its specification. There are various implementations of XML to JSON conversion tools, with

StAXON being one that works with Java XML libraries. The Apache Software Foundation offers Camel XMLJSON which follows very similar rules to StAXON for conversion[3].

From looking at a few conversion tools it appears that there is a semi-standard set of rules for how to do the conversion between XML and JSON that then various implementations of those rules.

REFERENCES:
[1]: http://docs.oracle.com/javase/tutorial/jaxp/stax/why.html
[2]: XML and JSON conversion rules:
https://github.com/beckchr/staxon/wiki/Mapping-Convention
[3]: Apache Camel XMLJSON:
http://camel.apache.org/xmljson.html

Question 3: Implement Java DOM and SAX parsers to display the contents of nodes a) item name with tag <Name> b) office location with tag<office> using file named sample.XML C) sold date (see the attached file).

See attached Java program.

Output of program in appendix.

Appendix:

```
run:
Here's the output using the DOM Parser:
Order:
        item name: Tech Track Jacket
        Office location: Widgets, Boston MA
        SoldDate: 8/01/2015
Order:
        item name: Twix Polo
        Office location: Widgets, Boston MA
        SoldDate: 9/01/2015
Order:
        item name: Barricade Bermuda
        Office location: Widgets, Boston MA
        SoldDate: 9/16/2015
Order:
        item name:  Tech Cape
        Office location: Widgets, Boston MA
        SoldDate: 9/14/2015
Order:
        item name: Shirt
        Office location: Widgets, Boston MA
        SoldDate: 9/19/2015
Order:
        item name: Barricade Bermuda
        Office location: Widgets, Boston MA
        SoldDate: 9/17/2015
Order:
        item name: Twix Polo
        Office location: Widgets, Boston MA
        SoldDate: 8/20/2015
Order:
        item name: Shirt
        Office location: Widgets, Boston MA
        SoldDate: 8/14/2015
Order:
        item name: Twix Polo
        Office location: Widgets, Boston MA
        SoldDate: 9/01/2015
Order:
        item name: Rally Funnel Neck Hoody
        Office location: Widgets, Boston MA
        SoldDate: 9/23/2015
```

BUILD SUCCESSFUL (total time: 0 seconds)

run:
Here's the output using the SAX Parser:
{Start Document}
Order:
item name: Tech Track Jacket
Office location: Widgets, Boston MA
SoldDate: 8/01/2015
Order:
item name: Twix Polo
Office location: Widgets, Boston MA
SoldDate: 9/01/2015
Order:
item name: Barricade Bermuda
Office location: Widgets, Boston MA
SoldDate: 9/16/2015
Order:
item name: Tech Cape
Office location: Widgets, Boston MA
SoldDate: 9/14/2015
Order:
item name: Shirt
Office location: Widgets, Boston MA
SoldDate: 9/19/2015
Order:
item name: Barricade Bermuda
Office location: Widgets, Boston MA
SoldDate: 9/17/2015
Order:
item name: Twix Polo
Office location: Widgets, Boston MA
SoldDate: 8/20/2015
Order:
item name: Shirt
Office location: Widgets, Boston MA
SoldDate: 8/14/2015
Order:
item name: Twix Polo
Office location: Widgets, Boston MA
SoldDate: 9/01/2015
Order:

item name: Rally Funnel Neck Hoody
Office location: Widgets, Boston MA
SoldDate: 9/23/2015
{End Document}
BUILD SUCCESSFUL (total time: 0 seconds)