Return to Module Overview Page (https://onlinelearning.berkeley.edu/courses/665040/pages/module-nine)

9.4: Java XML APIs

We have discussed how XML describes, structures and tags data. Also, we talked about what a well-formed, valid XML document is. With that in mind, we are now ready to parse and create XML documents using the Java XML APIs.

There are two major types of XML APIs in Java: Tree based APIs and event based APIs.

The Java DOM API is a tree based API that follows the DOM (document object model) representation. If you have programmed in Javascript, it will look familiar to you. The DOM parser works by reading an entire XML document into memory and builds a tree representation of its contents. You will only need to write code to instantiate a parser and pass it an XML document. The rest of the work is handled for you and is stored in built in DOM objects. As a result, less code is required, but at the cost of memory, processing time and flexibility.

Event based APIs include SAX (Simple API for XML) and StAX (Streaming API for XML). As parts of an XML document is read, events are generated based on the type of data. In DOM, the handlers are already written. In event based APIs, you must write these handlers. Although it requires more work, you gain plenty of flexibility. You can choose which specific data to process and you do not always have to read or store the entire file, which frees up resources.

When you want to map the XML data with your custom data structures, event based APIs are more suitable than the DOM API. In DOM, storing the data in your data structures would be redundant since they are already stored in a tree.

The table below shows a comparison between the APIs:

Table: Comparing APIs

	DOM API	**StAX**
Parser Type	Reads Entire file into memory	Streaming, event based
Document Reading	Entire Document	Stream
Storage	Built-in tree	User defined
Validation	YES	NO (as of this writing)
XML Writing	NO	YES

Note #1: StAX parsers are not required to be able to validate and the reference implementation unfortunately, cannot validate yet.

Note #2: Throughout this module, we use factory methods instead of new to instantiate parsers. This allows more flexibility. So instead of being limited to one implementation for creating parsers, it can be overridden and changed during run time. You can read more about them in the book, Design Patterns (by Gamma, Helm, et al.).