LIYANA ARACHCHIGE RANIL

SHORT NOTES / STAX vs SAX vs DOM

1. StAX stands for STREAMING API FOR XML
2. A streaming java based , event-driven , pull (client PULLS , **KAKULA (CLIENT->PULLS)**) parsing API for READING and WRITING XML documents
3. StAX enable you to create BIDIRECTIONAL XML parsers
4. StAX has only a LIGHT MEMORY FOOT PRINT
5. StAX provides STANDARD , BIDIRECTIONAL PULL PARSER for STREAMING XML processing
6. More efficient memory management than DOM
7. The major GOAL of StAX is “to give Parsing Control to the PRGRAMMER by EXPOSING a SIMPLE ITERATOR based API”
8. This allows the programmer to ASK the NEXT event (PULL EVENT) and allows state to be stored in procedural fashion
9. StAX was created to address the LIMITATIONS of two other parsers DOM and SAX
10. Streaming verses DOM – there are two types of programming models when working with XML infosets. STREAMING and DOCUMENT OBJECT MODEL
11. The DOM models involves creating IN-MEMORY objects representing an entire document tree and the complete infoset state for an XML document
12. Once in memory , DOM trees can be NAVIGATED freely and parsed ARBITARILY and provide MAXIMUM flexibility to PROGRAMMERS
13. However , the MEMORY FOOT PRINT and PROCESSING REQUIREMENTS are high for DOM
14. STREAMING refers to s PROGRAMMING MODEL in which XML infosets are TRANSMITTED and PARSED SERIALLY at application

RUNTIME. Often in REALTIME and often DYNAMIC source whose content are not PRECISELY KNOWN beforehand

15. STREAM based parsers can start generating output IMMEDIATEL and Infoset items can be discharged and garbage collected IMMEDIATELY after they are used
16. The primary trade off in STREAMING is you can only see the infoset state at one location at a time in the document
17. STREAMING model is useful when application has STRICT memory requirements
18. STREAMING PULL parsing verse STREAMING PUSH parsing – CLIENT APPLICATION CALLS methods on an XML parsing library when it NEEDS to interact with XML infoset , that is CLIENT PULLS
19. STREAMING PISH PARSING – XML parser SENDS (PUSHES) XML data to the client as the parser encounters element in XML
20. PULL parsing advantages over PUSH parsing –
    a. With PULL parsing CLIENT CONTROLS the application thread, PUSH parsing parser controls the application thread
    b. PULL parsing libraries can be mush SMALLER and the client code to interact with those libraries much SIMPLER that PUSH
    c. PULL clients can read MULTIPLE documents at one time with a SINGLE thread
    d. A StAX PULL parser **can FILTER XML** documents such that elements UNNECCESSARY to the client can be IGNORED
21. StAX use cases
    a. Data Binding
        i. Unmarshalling XML documents
        ii. Marshalling XML Documents
        iii. Parallel Document Processing
        iv. Wireless Communication

b. SOAP Message Processing
    i. Parsing simple predictable structures
    ii. Parsing GRAPH representation with forward references
    iii. Parsing WSDL
c. Virtual Data Sources
    i. Viewing XML data stored in data bases
    ii. Viewing data in java objects created by XML data binding
    iii. Navigating a DOM tree as a stream of events
d. Parsing specific XML vocabularies
e. Pipelined XML Processing

22. StAX is not as POWERFUL as TraX or JDOM , but neither does it REQUIRE as much memory or processor load to be useful
23. StAX enabled clients are generally EASIER to code than SAX clients
24. StAX is BIDIRECTIONAL , it can both READ and WRITE XML documents. SAX is READ ONLY
25. SAX is a PUSH API , StAX is a PULL API

26.

TABLE 18–1 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	Not supported	Not supported	Supported	Supported
CPU and Memory Efficiency	Good	Good	Varies	Varies
Forward Only	Supported	Supported	Not supported	Not supported
Read XML	Supported	Supported	Supported	Supported
Write XML	Supported	Not supported	Supported	Supported
Create, Read, Update, Delete	Not supported	Not supported	Supported	Not supported

27. StAX has two API , namely CURSOR API and ITERATOR API
28. Cursor API represents a CURSOR with which you can walk XML document from BEGINNING to END. Always move FORWARD ,never BACK WARD
29. XMLStreamReader , XMLStreamWriter are two main Cursor interfaces
30. ITERATOR API represents XML document stream as a set of DESCRETE EVENT OBJECTS
31. These events are PULLED by the application and PROVIDED by the parser in the order they are read in the source XML document

32. XMLEvent , XMLEventReader , XMLEventWriter are major interfaces