System Development Process:

Large File Management System

Documentation:

Made by Sanket Bhimani, Meet Dave and Dhara Ka. patel on date 26/12/2016 version 0.1

Information about SAX parser:

SAX (the Simple API for XML) is an event-based parser for xml documents.Unlike a DOM parser, SAX parser creates no parse tree.

Characteristics of SAX parser:

* API type: push, streaming
* Easy to use
* CPU and memory efficient
* Forward only
* Read only

It works in below steps:

1. Reads an XML document from top to bottom, recognizing the tokens that make up a well-formed XML document.
2. Tokens are processed in the same order that they appear in the document
3. Reports the application program the nature of tokens that the parser has encountered as they occur.
4. The application program provides an "event" handler that must be registered with the parser
5. As the tokens are identified, callback methods in the handler are invoked with the relevant information

SAX parser is used when:

1. The file is too big to be loaded in main memory. DOM parser takes much memory to build the tree.
2. The problem to be solved involves only part of the XML document
3. Data is available as soon as it is seen by the parser, so SAX works well for an XML document that arrives over a stream.

Disadvantages of using SAX:

1. We have no random access to an XML document since it is processed in a forward-only manner
2. If you need to keep track of data the parser has seen or change the order of items, you must write the code and store the data on your own

Methods used:

1. *void startElement(String uri, String localName, String qName, Attributes atts)*
   * Called at the beginning of an element.
2. *void endElement(String uri, String localName,String qName)*
   * Called at the end of an element.
3. *public void parse(InputSource source)*
   * Parse an XML document.

The application can use this method to instruct the SAX parser to begin parsing an XML document from any valid input source (A character stream, a byte stream, or a URI).

**Parameters:** source - The input source for the top-level of the XML document.

Example:public interface XMLStreamReader {

public int next() throws XMLStreamException;

public boolean hasNext() throws XMLStreamException;

public String getText();

public String getLocalName();

public String getNamespaceURI();

// ...other methods not shown

}

This example illustrates how to instantiate an input factory, create a reader and iterate over the elements of an XML document.

This example illustrates how to instantiate an input factory, create a reader and iterate over the elements of an XML document.

Example: package xml.sax;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

import java.util.\*;

public class SaxHandler extends DefaultHandler {

public List<Driver> drivers = new ArrayList<Driver>();

public Map<String, Vehicle> vehicles = new HashMap<String, Vehicle>();

private Stack<String> elementStack = new Stack<String>();

private Stack<Object> objectStack = new Stack<Object>();

public void startElement(String uri, String localName,

String qName, Attributes attributes) throws SAXException {

//code

}

public void endElement(String uri, String localName,

String qName) throws SAXException {

//code

}

private String currentElement() {

//code

}

private String currentElementParent() {

//code

}

}

**Example:** public class SaxParserExample {

public static void main (String argv []) {

SAXParserFactory factory = SAXParserFactory.newInstance();

try {

InputStream xmlInput =

new FileInputStream("data\\sax-example.xml");

SAXParser saxParser = factory.newSAXParser();

SaxHandler handler = new SaxHandler();

saxParser.parse(xmlInput, handler);

for(Driver driver : handler.drivers){

System.out.println(driver);

}

} catch (Throwable err) {

err.printStackTrace ();

}

}

}

How to create and use XML file:

* Microsoft XML notepad is an application that allows you to create and edit XML document easily. Create new document in XML notepad. Write XML code. To view document source go in menu View and Source.
* Notepad can also be used to create XML file. Save file with ‘.xml’ format.

You can’t run XML file itself. It is a data format not a programming language. Use XML file in other programming language code and it will give application specific information.

How to create java file:

See the code attached with this document which allows you to search definition of given word and also allows you to search related definition.

StAX Parser:

Streaming API for XML (StAX) is an application programming interface (API) to read and write XML documents, originating from the Java programming language community.

StAX is vaguely presented as the middle between DOM and SAX: it's easier than SAX and more efficient than DOM.

It is a PULL model.

StAX parser used when:

1. You can process the XML document in a linear fashion from the top down.
2. The document is not deeply nested.
3. You are processing a very large XML document whose DOM tree would consume too much memory. Typical DOM implementations use ten bytes of memory to represent one byte of XML.
4. The problem to be solved involves only part of the XML document.
5. Data is available as soon as it is seen by the parser, so StAX works well for an XML document that arrives over a stream.

Disadvantages of using StAX parser:

1. We have no random access to an XML document since it is processed in a forward-only manner.

2.If you need to keep track of data the parser has seen or change the order of items, you must write the code and store the data on your own.

XMLStreamReader Class:

This class provide iterator of events which can be used to iterate over events as they occur while parsing the XML document

1. int next() - used to retrieve next event.
2. boolean hasNext() - used to check further events exists or not
3. String getText() - used to get text of an element
4. String getLocalName() - used to get name of an element

Example: public interface XMLStreamReader {

public int next() throws XMLStreamException;

public boolean hasNext() throws XMLStreamException;

public String getText();

public String getLocalName();

public String getNamespaceURI();

// ...other methods not shown

}

This example illustrates how to instantiate an input factory, create a reader and iterate over the elements of an XML document.

XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();

XMLStreamReader xmlStreamReader = xmlInputFactory.createXMLStreamReader(...);

while (xmlStreamReader.hasNext())

{

xmlStreamReader.next();

}

XMLStreamConstants

It is a public interface.This interface declares the constants used in this API.

1.ATTRIBUTE

Indicates an event is an attribute

2.END\_DOCUMENT

Indicates an event is an end document

3.END\_ELEMENT

Indicates an event is an end element

4.START\_DOCUMENT

Indicates an event is a start document

5.START\_ELEMENT

Indicates an event is a start element

Example: private void skipTo(String[] startElementNames, String[] stopElementNames) throws XMLStreamException {

boolean found = false ;

loop : while (parser.hasNext()) {

int event = parser.next() ;

switch (event) {

case XMLStreamConstants.END\_DOCUMENT :

break loop ;

case XMLStreamConstants.END\_ELEMENT :

if ( stopElementNames == null )

//code

if ( endTag != null && containsName(stopElementNames, endTag)

//code

break ;

case XMLStreamConstants.START\_ELEMENT :

if ( startElementNames == null )

break ;

// code

}

}

Class XMLInputFactory

Defines an abstract implementation of a factory for getting streams

1.static XMLInputFactory newInstance()

-Creates a new instance of the factory in exactly the same manner as the newFactory() method.

2.abstract XMLStreamReader createXMLStreamReader(InputStream stream)

-Create a new XMLStreamReader from a java.io.InputStream

Example:

XMLInputFactory factory = XMLInputFactory.newInstance();

XMLEventReader eventReader =

factory.createXMLEventReader(

new FileReader("data\\test.xml"));

XMLStreamReader streamReader =

factory.createXMLStreamReader(

new FileReader("data\\test.xml"));

Word Search software:

-issues:

1.Large xml file management

2. Memory stack overflow

3. Large processing time

Basic introduction:

In word search project user enters a word.The system finds all relevent definations and Hyponyms and Hypernyms relates to that defination.

If user left clicks the node ,node is changed.and if user right clicks the node the node is expanded.

Implementation basics:

For implementation of graph jit charts Hyper tree is used.The data is stored in xml file. In ENWsynset.xml the word and its Id is stored. In ENWlexicalEntry.xml the Id ,definition and the relations is stored. Search.java file finds the for the search key in ENWlexicalEntry.xml for the word Id and stores in a 'synset' array. Then the definition is found from ENWsynset.xml file and stored into 'definition' array.The related word's Id are stored in 'hypernym' and 'hyponym' array and related word definitions are found from ENWlexicalEntry.xml. Searchprocess.java file makes json from the dynamic arrays.

File aaa.html uses the Hyper tree methods to represent the json data.

Example of JSON response:

[

{

"id": "0hypr",

"name": "degree",

"adjacencies": [],

"data": {

"$color": "#0000FF",

"$type": "square",

"$dim": 7

}

},

{

"id": 1,

"name": "the furthest or highest degree of something",

"adjacencies": [

{

"nodeTo": "0hypr",

"nodeFrom": 1,

"data": {

"$color": "#0000FF"

}

}

],

"data": {

"$color": "#000000",

"$type": "square",

"$dim": 7

}

},

{

"id": "1hypr",

"name": "extremity",

"adjacencies": [],

"data": {

"$color": "#0000FF",

"$type": "square",

"$dim": 7

}

},

{

"id": "2hypo",

"name": "acme",

"adjacencies": [],

"data": {

"$color": "#FF0000",

"$type": "square",

"$dim": 7

}

},

{

"id": 3,

"name": "the point located farthest from the middle of something ",

"adjacencies": [

{

"nodeTo": "1hypr",

"nodeFrom": 3,

"data": {

"$color": "#0000FF"

}

},

{

"nodeTo": "2hypo",

"nodeFrom": 3,

"data": {

"$color": "#0000FF"

}

}

],

"data": {

"$color": "#000000",

"$type": "square",

"$dim": 7

}

}

]

Future scope:

-zoom in -zoom out facility can be added for better user interface.

Faults:

1.Takes more time for processing of large file.

2. Needs more better user interaction

Revised work:

Force Directed Graph:

Implementation basics:

* The graph implementation is done in JIT Force directed graph.
* File temp.jsp provides basic user interaction for taking a searchkey input and loader. When the search button is pressed the search function is called.
* File search.java implements the search() function. That method finds the searchkey and stores the definition and its hyponyms –hypernyms in one hash map. It uses STAX parser for finding details from large XML file.
* File searchprocss.java calls the search process method and stores the different arrays for different definition of word. It also creates a Json response from stored hashmap and arrays.
* Sidepanel which shows all hyponyms and hyponyms words for each definitions.
* When user clicks on any word in side panel graph of that word will be displayed.