# 1. What is Document Object Model (DOM)

The Document Object Model (DOM) uses nodes to represent the HTML or XML document as a tree structure.

Below is a simple XML document:

```
<company>
  <staff id="1001">
    <firstname>yong</firstname>
    <lastname>mook kim</lastname>
    <nickname>mkyong</nickname>
    <salary currency="USD">100000</salary>
  </staff>
</company>
```

DOM common terms.

- The <company> is the root element.
- The <staff>, <firstname> and all <?> are the element nodes.
- The text node is the value wrapped by the element nodes; for example, <firstname>yong</firstname>, the yong is the text node.
- The attribute is part of the element node; for example, <staff id="1001"> the id is the attribute of the staff element.

# 2. Read or Parse a XML file

This example shows you how to use the Java built-in DOM parser APIs to read or parse an XML file.

2.1 Review below XML file.

`/users/mkyong/staff.xml`

```
<?xml version="1.0"?>
<company>
  <staff id="1001">
    <firstname>yong</firstname>
    <lastname>mook kim</lastname>
    <nickname>mkyong</nickname>
    <salary currency="USD">100000</salary>
  </staff>
  <staff id="2001">
    <firstname>low</firstname>
    <lastname>yin fong</lastname>
```

```
    <nickname>fong fong</nickname>
    <salary currency="INR">200000</salary>
  </staff>
</company>
```

2.2 Below is a DOM parser example of parsing or reading the above XML file.

`ReadXmlDomParser.java`

```
package com.mkyong.xml.dom;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;

public class ReadXmlDomParser {

  private static final String FILENAME = "/users/mkyong/staff.xml";

  public static void main(String[] args) {

    // Instantiate the Factory
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

    try {

        // optional, but recommended
        // process XML securely, avoid attacks like XML External Entities (XXE)
        dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, true);

        // parse XML file
        DocumentBuilder db = dbf.newDocumentBuilder();

        Document doc = db.parse(new File(FILENAME));

        // optional, but recommended
        // http://stackoverflow.com/questions/13786607/normalization-in-dom-parsing-with-java-how-does-it-work
        doc.getDocumentElement().normalize();

        System.out.println("Root Element :" + doc.getDocumentElement().getNodeName());
        System.out.println("------");

        // get <staff>
        NodeList list = doc.getElementsByTagName("staff");

        for (int temp = 0; temp < list.getLength(); temp++) {

            Node node = list.item(temp);
```

```
            if (node.getNodeType() == Node.ELEMENT_NODE) {

                Element element = (Element) node;

                // get staff's attribute
                String id = element.getAttribute("id");

                // get text
                String firstname = element.getElementsByTagName("firstname").item(0).getTextContent();
                String lastname = element.getElementsByTagName("lastname").item(0).getTextContent();
                String nickname = element.getElementsByTagName("nickname").item(0).getTextContent();

                NodeList salaryNodeList = element.getElementsByTagName("salary");
                String salary = salaryNodeList.item(0).getTextContent();

                // get salary's attribute
                String currency =
salaryNodeList.item(0).getAttributes().getNamedItem("currency").getTextContent();

                System.out.println("Current Element :" + node.getNodeName());
                System.out.println("Staff Id : " + id);
                System.out.println("First Name : " + firstname);
                System.out.println("Last Name : " + lastname);
                System.out.println("Nick Name : " + nickname);
                System.out.printf("Salary [Currency] : %,.2f [%s]%n%n", Float.parseFloat(salary), currency);

            }
        }

    } catch (ParserConfigurationException | SAXException | IOException e) {
        e.printStackTrace();
    }

  }

}
```

Output

```
Terminal

Root Element :company
------
Current Element :staff
Staff Id : 1001
First Name : yong
Last Name : mook kim
Nick Name : mkyong
Salary [Currency] : 100,000.00 [USD]

Current Element :staff
Staff Id : 2001
First Name : low
Last Name : yin fong
Nick Name : fong fong
Salary [Currency] : 200,000.00 [INR]
```

# 3. Read or Parse XML file (Unicode)

In DOM parser, there is no difference between reading a normal and Unicode XML file.

3.1 Review below XML file containing some Chinese characters (Unicode).

`src/main/resources/staff-unicode.xml`

```
<?xml version="1.0"?>
<company>
   <staff id="1001">
      <firstname>揚</firstname>
      <lastname>木金</lastname>
      <nickname>mkyong</nickname>
      <salary currency="USD">100000</salary>
   </staff>
   <staff id="2001">
      <firstname>low</firstname>
      <lastname>yin fong</lastname>
      <nickname>fong fong</nickname>
      <salary currency="INR">200000</salary>
   </staff>
</company>
```

3.2 The below example parse the above XML file; it loops all the nodes one by one and prints it out.

`ReadXmlDomParserLoop.java`

```
package com.mkyong.xml.dom;

import org.w3c.dom.*;
import org.xml.sax.SAXException;

import javax.xml.XMLConstants;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.IOException;
import java.io.InputStream;

public class ReadXmlDomParserLoop {

  public static void main(String[] args) {

     // Instantiate the Factory
     DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

     try (InputStream is = readXmlFileIntoInputStream("staff-unicode.xml")) {

        // parse XML file
        DocumentBuilder db = dbf.newDocumentBuilder();

        // read from a project's resources folder
        Document doc = db.parse(is);

        System.out.println("Root Element :" + doc.getDocumentElement().getNodeName());
        System.out.println("------");
```

```
        if (doc.hasChildNodes()) {
            printNote(doc.getChildNodes());
        }

    } catch (ParserConfigurationException | SAXException | IOException e) {
        e.printStackTrace();
    }

  }

  private static void printNote(NodeList nodeList) {

    for (int count = 0; count < nodeList.getLength(); count++) {

        Node tempNode = nodeList.item(count);

        // make sure it's element node.
        if (tempNode.getNodeType() == Node.ELEMENT_NODE) {

            // get node name and value
            System.out.println("\nNode Name =" + tempNode.getNodeName() + " [OPEN]");
            System.out.println("Node Value =" + tempNode.getTextContent());

            if (tempNode.hasAttributes()) {

                // get attributes names and values
                NamedNodeMap nodeMap = tempNode.getAttributes();
                for (int i = 0; i < nodeMap.getLength(); i++) {
                    Node node = nodeMap.item(i);
                    System.out.println("attr name : " + node.getNodeName());
                    System.out.println("attr value : " + node.getNodeValue());
                }

            }

            if (tempNode.hasChildNodes()) {
                // loop again if has child nodes
                printNote(tempNode.getChildNodes());
            }

            System.out.println("Node Name =" + tempNode.getNodeName() + " [CLOSE]");

        }

    }

  }

  // read file from project resource's folder.
  private static InputStream readXmlFileIntoInputStream(final String fileName) {
    return ReadXmlDomParserLoop.class.getClassLoader().getResourceAsStream(fileName);
  }

}
```

Output

```
Terminal
```

```
Root Element :company
------

Node Name =company [OPEN]
Node Value =

    揚
    木金
    mkyong
    100000


    low
    yin fong
    fong fong
    200000



Node Name =staff [OPEN]
Node Value =
    揚
    木金
    mkyong
    100000

attr name : id
attr value : 1001

Node Name =firstname [OPEN]
Node Value =揚
Node Name =firstname [CLOSE]

Node Name =lastname [OPEN]
Node Value =木金
Node Name =lastname [CLOSE]

Node Name =nickname [OPEN]
Node Value =mkyong
Node Name =nickname [CLOSE]

Node Name =salary [OPEN]
Node Value =100000
attr name : currency
attr value : USD
Node Name =salary [CLOSE]
Node Name =staff [CLOSE]

Node Name =staff [OPEN]
Node Value =
    low
    yin fong
    fong fong
    200000

attr name : id
attr value : 2001
```

```
Node Name =firstname [OPEN]
Node Value =low
Node Name =firstname [CLOSE]

Node Name =lastname [OPEN]
Node Value =yin fong
Node Name =lastname [CLOSE]

Node Name =nickname [OPEN]
Node Value =fong fong
Node Name =nickname [CLOSE]

Node Name =salary [OPEN]
Node Value =200000
attr name : currency
attr value : INR
Node Name =salary [CLOSE]
Node Name =staff [CLOSE]
Node Name =company [CLOSE]
```

# 4. Parse Alexa API XML Response

This example shows how to use the DOM parser to parse the XML response from Alexa's API.

4.1 Send a request to the following Alexa API.

`Terminal`

```
https://data.alexa.com/data?cli=10&url=mkyong.com
```

4.2 The Alexa API will return the following XML response. The Alexa ranking is inside the POPULARITY element, the TEXT attribute.

```
<!--  Need more Alexa data?  Find our APIs here: https://aws.amazon.com/alexa/  -->
<ALEXA VER="0.9" URL="mkyong.com/" HOME="0" AID="=" IDN="mkyong.com/">
 <SD>
  <POPULARITY URL="mkyong.com/" TEXT="20162" SOURCE="panel"/>
  <REACH RANK="14430"/>
  <RANK DELTA="+947"/>
  <COUNTRY CODE="IN" NAME="India" RANK="4951"/>
 </SD>
</ALEXA>
```

4.3 We use a DOM parser to directly select the POPULARITY element and print out the value of the TEXT attribute.

`ReadXmlAlexaApi.java`

```
package com.mkyong.xml.dom;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
```

```
import org.w3c.dom.NodeList;

import javax.xml.XMLConstants;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;

public class ReadXmlAlexaApi {

   private static final String ALEXA_API = "http://data.alexa.com/data?cli=10&url=";
   private final DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

   public static void main(String[] args) {

      ReadXmlAlexaApi obj = new ReadXmlAlexaApi();
      int alexaRanking = obj.getAlexaRanking("mkyong.com");

      System.out.println("Ranking: " + alexaRanking);

   }

   public int getAlexaRanking(String domain) {

      int result = 0;

      String url = ALEXA_API + domain;

      try {

         URLConnection conn = new URL(url).openConnection();

         try (InputStream is = conn.getInputStream()) {

            // unknown XML better turn on this
            dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, true);

            DocumentBuilder dBuilder = dbf.newDocumentBuilder();

            Document doc = dBuilder.parse(is);

            Element element = doc.getDocumentElement();

            // find this tag "POPULARITY"
            NodeList nodeList = element.getElementsByTagName("POPULARITY");
            if (nodeList.getLength() > 0) {

               Element elementAttribute = (Element) nodeList.item(0);
               String ranking = elementAttribute.getAttribute("TEXT");
               if (!"".equals(ranking)) {
                  result = Integer.parseInt(ranking);
               }
            }
         }

      } catch (Exception e) {
         e.printStackTrace();
         throw new IllegalArgumentException("Invalid request for domain : " + domain);
```

```
        }

        return result;
    }

}
```

The domain mkyong.com ranked 20162.

```
Terminal

Ranking: 20162
```

# Read or Parse a XML file (SAX)

This example shows you how to use the Java built-in SAX parser APIs to read or parse an XML file.

2.1 Below is an XML file.

src/main/resources/staff.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Company>
    <staff id="1001">
        <name>mkyong</name>
        <role>support</role>
        <salary currency="USD">5000</salary>
        <!-- for special characters like < &, need CDATA -->
        <bio><![CDATA[HTML tag <code>testing</code>]]></bio>
    </staff>
    <staff id="1002">
        <name>yflow</name>
        <role>admin</role>
        <salary currency="EUR">8000</salary>
        <bio><![CDATA[a & b]]></bio>
    </staff>
</Company>
```

*P.S In the XML file, for those special characters like < or &, we need to wrap it with CDATA.*

2.2 Create a class to extend org.xml.sax.helpers.DefaultHandler, and override the startElement, endElement and characters methods to print all the XML elements, attributes, comments and texts.

PrintAllHandlerSax.java

```
package com.mkyong.xml.sax.handler;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
```

```
public class PrintAllHandlerSax extends DefaultHandler {

  private StringBuilder currentValue = new StringBuilder();

  @Override
  public void startDocument() {
     System.out.println("Start Document");
  }

  @Override
  public void endDocument() {
     System.out.println("End Document");
  }

  @Override
  public void startElement(
        String uri,
        String localName,
        String qName,
        Attributes attributes) {

     // reset the tag value
     currentValue.setLength(0);

     System.out.printf("Start Element : %s%n", qName);

     if (qName.equalsIgnoreCase("staff")) {
        // get tag's attribute by name
        String id = attributes.getValue("id");
        System.out.printf("Staff id : %s%n", id);
     }

     if (qName.equalsIgnoreCase("salary")) {
        // get tag's attribute by index, 0 = first attribute
        String currency = attributes.getValue(0);
        System.out.printf("Currency :%s%n", currency);
     }

  }

  @Override
  public void endElement(String uri,
                  String localName,
                  String qName) {

     System.out.printf("End Element : %s%n", qName);

     if (qName.equalsIgnoreCase("name")) {
        System.out.printf("Name : %s%n", currentValue.toString());
     }

     if (qName.equalsIgnoreCase("role")) {
        System.out.printf("Role : %s%n", currentValue.toString());
     }

     if (qName.equalsIgnoreCase("salary")) {
        System.out.printf("Salary : %s%n", currentValue.toString());
     }
```

```
    if (qName.equalsIgnoreCase("bio")) {
        System.out.printf("Bio : %s%n", currentValue.toString());
    }

  }

  //
http://www.saxproject.org/apidoc/org/xml/sax/ContentHandler.html#characters%28char%5B%5D,%20int,%20int%29
  // SAX parsers may return all contiguous character data in a single chunk,
  // or they may split it into several chunks
  @Override
  public void characters(char ch[], int start, int length) {

    // The characters() method can be called multiple times for a single text node.
    // Some values may missing if assign to a new string

    // avoid doing this
    // value = new String(ch, start, length);

    // better append it, works for single or multiple calls
    currentValue.append(ch, start, length);

  }

}
```

2.3 SAXParser to parse an XML file.

ReadXmlSaxParser.java

```
package com.mkyong.xml.sax;

import com.mkyong.xml.sax.handler.PrintAllHandlerSax;
import org.xml.sax.SAXException;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import java.io.IOException;

public class ReadXmlSaxParser {

    private static final String FILENAME = "src/main/resources/staff.xml";

    public static void main(String[] args) {

        SAXParserFactory factory = SAXParserFactory.newInstance();

        try {

            // XXE attack, see https://rules.sonarsource.com/java/RSPEC-2755
            SAXParser saxParser = factory.newSAXParser();

            PrintAllHandlerSax handler = new PrintAllHandlerSax();

            saxParser.parse(FILENAME, handler);

        } catch (ParserConfigurationException | SAXException | IOException e) {
```

```
            e.printStackTrace();
        }

    }

}
```

Output

```
Terminal
```

```
Start Document
Start Element : Company
Start Element : staff
Staff id : 1001
Start Element : name
End Element : name
Name : mkyong
Start Element : role
End Element : role
Role : support
Start Element : salary
Currency :USD
End Element : salary
Salary : 5000
Start Element : bio
End Element : bio
Bio : HTML tag <code>testing</code>
End Element : staff
Start Element : staff
Staff id : 1002
Start Element : name
End Element : name
Name : yflow
Start Element : role
End Element : role
Role : admin
Start Element : salary
Currency :EUR
End Element : salary
Salary : 8000
Start Element : bio
End Element : bio
Bio : a & b
End Element : staff
End Element : Company
End Document
```

```
 SAXParserFactory factory = SAXParserFactory.newInstance();

 try {

    // https://rules.sonarsource.com/java/RSPEC-2755
    // prevent XXE, completely disable DOCTYPE declaration:
    factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);

    SAXParser saxParser = factory.newSAXParser();
```

```
    PrintAllHandlerSax handler = new PrintAllHandlerSax();

    saxParser.parse(FILENAME, handler);

 } catch (ParserConfigurationException | SAXException | IOException e) {
    e.printStackTrace();
 }
```

# 3. Convert an XML file to an object

This example parses an XML file and converts it into a List of objects. It works, but not recommended, try JAXB

3.1 Review the same XML file.

src/main/resources/staff.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Company>
   <staff id="1001">
      <name>mkyong</name>
      <role>support</role>
      <salary currency="USD">5000</salary>
      <!-- for special characters like < &, need CDATA -->
      <bio><![CDATA[HTML tag <code>testing</code>]]></bio>
   </staff>
   <staff id="1002">
      <name>yflow</name>
      <role>admin</role>
      <salary currency="EUR">8000</salary>
      <bio><![CDATA[a & b]]></bio>
   </staff>
</Company>
```

3.2 And we want to convert the above XML file into the following Staff object.

Staff.java

```
package com.mkyong.xml.sax.model;

import java.math.BigDecimal;

public class Staff {

  private Long id;
  private String name;
  private String role;
  private BigDecimal salary;
  private String Currency;
  private String bio;

  //... getters, setters...toString
}
```

3.3 The below class will do the XML to Object conversion.

MapStaffObjectHandlerSax.java

```
package com.mkyong.xml.sax.handler;

import com.mkyong.xml.sax.model.Staff;
import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;

public class MapStaffObjectHandlerSax extends DefaultHandler {

   private StringBuilder currentValue = new StringBuilder();
   List<Staff> result;
   Staff currentStaff;

   public List<Staff> getResult() {
      return result;
   }

   @Override
   public void startDocument() {
      result = new ArrayList<>();
   }

   @Override
   public void startElement(
         String uri,
         String localName,
         String qName,
         Attributes attributes) {

      // reset the tag value
      currentValue.setLength(0);

      // start of loop
      if (qName.equalsIgnoreCase("staff")) {

         // new staff
         currentStaff = new Staff();

         // staff id
         String id = attributes.getValue("id");
         currentStaff.setId(Long.valueOf(id));
      }

      if (qName.equalsIgnoreCase("salary")) {
         // salary currency
         String currency = attributes.getValue("currency");
         currentStaff.setCurrency(currency);
      }

   }

   public void endElement(String uri,
                          String localName,
                          String qName) {
```

```
        if (qName.equalsIgnoreCase("name")) {
            currentStaff.setName(currentValue.toString());
        }

        if (qName.equalsIgnoreCase("role")) {
            currentStaff.setRole(currentValue.toString());
        }

        if (qName.equalsIgnoreCase("salary")) {
            currentStaff.setSalary(new BigDecimal(currentValue.toString()));
        }

        if (qName.equalsIgnoreCase("bio")) {
            currentStaff.setBio(currentValue.toString());
        }

        // end of loop
        if (qName.equalsIgnoreCase("staff")) {
            result.add(currentStaff);
        }

    }

    public void characters(char ch[], int start, int length) {
        currentValue.append(ch, start, length);

    }

}
```

3.4 Run it.

ReadXmlSaxParser2.java

```
package com.mkyong.xml.sax;

import com.mkyong.xml.sax.handler.MapStaffObjectHandlerSax;
import com.mkyong.xml.sax.model.Staff;
import org.xml.sax.SAXException;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

public class ReadXmlSaxParser2 {

    public static void main(String[] args) {

        SAXParserFactory factory = SAXParserFactory.newInstance();

        try (InputStream is = getXMLFileAsStream()) {

            SAXParser saxParser = factory.newSAXParser();

            // parse XML and map to object, it works, but not recommend, try JAXB
```

```
        MapStaffObjectHandlerSax handler = new MapStaffObjectHandlerSax();

        saxParser.parse(is, handler);

        // print all
        List<Staff> result = handler.getResult();
        result.forEach(System.out::println);

      } catch (ParserConfigurationException | SAXException | IOException e) {
        e.printStackTrace();
      }

    }

    // get XML file from resources folder.
    private static InputStream getXMLFileAsStream() {
      return ReadXmlSaxParser2.class.getClassLoader().getResourceAsStream("staff.xml");
    }

}
```

Output

```
Terminal

Staff{id=1001, name='揚木金', role='support', salary=5000, Currency='USD', bio='HTML tag
<code>testing</code>'}
Staff{id=1002, name='yflow', role='admin', salary=8000, Currency='EUR', bio='a & b'}
```

# 4. SAX Error Handler

This example shows how to register a custom error handler for the SAX parser.

4.1 Create a class and extends org.xml.sax.ErrorHandler. Read the code for self-explanation. It just wrapped the originate error message.

```
CustomErrorHandlerSax.java

package com.mkyong.xml.sax.handler;

import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

import java.io.PrintStream;

public class CustomErrorHandlerSax implements ErrorHandler {

    private PrintStream out;

    public CustomErrorHandlerSax(PrintStream out) {
      this.out = out;
    }

    private String getParseExceptionInfo(SAXParseException spe) {
      String systemId = spe.getSystemId();
```

```
        if (systemId == null) {
            systemId = "null";
        }

        String info = "URI=" + systemId + " Line="
                + spe.getLineNumber() + ": " + spe.getMessage();

        return info;
    }

    public void warning(SAXParseException spe) throws SAXException {
        out.println("Warning: " + getParseExceptionInfo(spe));
    }

    public void error(SAXParseException spe) throws SAXException {
        String message = "Error: " + getParseExceptionInfo(spe);
        throw new SAXException(message);
    }

    public void fatalError(SAXParseException spe) throws SAXException {
        String message = "Fatal Error: " + getParseExceptionInfo(spe);
        throw new SAXException(message);
    }

}
```

4.2 We use saxParser.getXMLReader() to get a org.xml.sax.XMLReader, it provide more options to configure the SAX parser.

ReadXmlSaxParser3.java

```
package com.mkyong.xml.sax;

import com.mkyong.xml.sax.handler.CustomErrorHandlerSax;
import com.mkyong.xml.sax.handler.MapStaffObjectHandlerSax;
import com.mkyong.xml.sax.model.Staff;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

public class ReadXmlSaxParser3 {

  public static void main(String[] args) {

    SAXParserFactory factory = SAXParserFactory.newInstance();

    try (InputStream is = getXMLFileAsStream()) {

        SAXParser saxParser = factory.newSAXParser();

        // parse XML and map to object, it works, but not recommend, try JAXB
```

```
        MapStaffObjectHandlerSax handler = new MapStaffObjectHandlerSax();

        // try XMLReader
        //saxParser.parse(is, handler);

        // more options for configuration
        XMLReader xmlReader = saxParser.getXMLReader();

        // set our custom error handler
        xmlReader.setErrorHandler(new CustomErrorHandlerSax(System.err));

        xmlReader.setContentHandler(handler);

        InputSource source = new InputSource(is);

        xmlReader.parse(source);

        // print all
        List<Staff> result = handler.getResult();
        result.forEach(System.out::println);

    } catch (ParserConfigurationException | SAXException | IOException e) {
        e.printStackTrace();
    }

 }

 // get XML file from resources folder.
 private static InputStream getXMLFileAsStream() {
    return ReadXmlSaxParser2.class.getClassLoader().getResourceAsStream("staff.xml");
 }

}
```

4.3 Update the staff.xml, remove the CDATA in the bio element, and put a &, and the SAX parser will hit an error.

src/main/resources/staff.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Company>
   <staff id="1001">
      <name>mkyong</name>
      <role>support</role>
      <salary currency="USD">5000</salary>
      <!-- for special characters like < &, need CDATA -->
      <bio>&</bio>
   </staff>
</Company>
```

4.4 Run it with the above custom error handler.

```
xmlReader.setErrorHandler(new CustomErrorHandlerSax(System.err));
```

Output

Terminal

```
org.xml.sax.SAXException: Fatal Error: URI=null Line=8: The entity name must immediately follow the '&' in the entity reference.
at com.mkyong.xml.sax.handler.CustomErrorHandlerSax.fatalError(CustomErrorHandlerSax.java:41)
at java.xml/com.sun.org.apache.xerces.internal.util.ErrorHandlerWrapper.fatalError(ErrorHandlerWrapper.java:181)
at java.xml/com.sun.org.apache.xerces.internal.impl.XMLErrorReporter.reportError(XMLErrorReporter.java:400)
at java.xml/com.sun.org.apache.xerces.internal.impl.XMLErrorReporter.reportError(XMLErrorReporter.java:327)
at java.xml/com.sun.org.apache.xerces.internal.impl.XMLScanner.reportFatalError(XMLScanner.java:1471)
//...
```

4.5 Run it without a custom error handler.

```
// xmlReader.setErrorHandler(new CustomErrorHandlerSax(System.err));
```

Output

```
Terminal

[Fatal Error] :8:15: The entity name must immediately follow the '&' in the entity reference.
org.xml.sax.SAXParseException; lineNumber: 8; columnNumber: 15; The entity name must immediately follow the '&' in the entity reference.
        at java.xml/com.sun.org.apache.xerces.internal.parsers.AbstractSAXParser.parse(AbstractSAXParser.java:1243)
        at java.xml/com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser.parse(SAXParserImpl.java:635)
        at com.mkyong.xml.sax.ReadXmlSaxParser2.main(ReadXmlSaxParser2.java:44)
```

# 5. SAX and Unicode

For XML files containing Unicode characters, by default, SAX can follow the XML encoding (default UTF-8) and parse the content correctly.

5.1 We can define the encoding at the top of the XML file, encoding="encoding-code"; for example, below is an XML file using the UTF-8 encoding.

```
<?xml version="1.0" encoding="utf-8"?>
<Company>
  <staff id="1001">
    <name>揚木金</name>
    <role>support</role>
    <salary currency="USD">5000</salary>
    <bio><![CDATA[HTML tag <code>testing</code>]]></bio>
  </staff>
  <staff id="1002">
    <name>yflow</name>
    <role>admin</role>
```

```
    <salary currency="EUR">8000</salary>
    <bio><![CDATA[a & b]]></bio>
  </staff>
</Company>
```

5.2 Alternatively, we can define a specified encoding in the InputSource.

```
XMLReader xmlReader = saxParser.getXMLReader();
 xmlReader.setContentHandler(handler);

 InputSource source = new InputSource(is);

 // set encoding
 source.setEncoding(StandardCharsets.UTF_8.toString());

 //source.setEncoding(StandardCharsets.UTF_16.toString());

 xmlReader.parse(source);
```

## How to parse JSON in Java

JSON (JavaScript Object Notation) is a lightweight, text-based, language-independent data exchange format that is easy for humans and machines to read and write. JSON can represent two structured types: objects and arrays. An object is an unordered collection of zero or more name/value pairs. An array is an ordered sequence of zero or more values. The values can be strings, numbers, booleans, null, and these two structured types.
Below is a simple example from Wikipedia that shows JSON representation of an object that describes a person. The object has string values for first name and last name, a number value for age, an object value representing the person's address, and an array value of phone number objects.

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        {
            "type": "home",
```

```
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

**Write JSON to a file**

Let us see an example that writes above JSON data into a file "JSONExample.json", with help of JSONObject and JSONArray.

```
// Java program for write JSON to a file


import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.LinkedHashMap;
import java.util.Map;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;


public class JSONWriteExample
{
    public static void main(String[] args) throws FileNotFoundException
    {
        // creating JSONObject
        JSONObject jo = new JSONObject();


        // putting data to JSONObject
        jo.put("firstName", "John");
        jo.put("lastName", "Smith");
        jo.put("age", 25);
```

```
// for address data, first create LinkedHashMap
Map m = new LinkedHashMap(4);
m.put("streetAddress", "21 2nd Street");
m.put("city", "New York");
m.put("state", "NY");
m.put("postalCode", 10021);

// putting address to JSONObject
jo.put("address", m);

// for phone numbers, first create JSONArray
JSONArray ja = new JSONArray();

m = new LinkedHashMap(2);
m.put("type", "home");
m.put("number", "212 555-1234");

// adding map to list
ja.add(m);

m = new LinkedHashMap(2);
m.put("type", "fax");
m.put("number", "212 555-1234");

// adding map to list
ja.add(m);

// putting phoneNumbers to JSONObject
jo.put("phoneNumbers", ja);
```

```
        // writing JSON to file:"JSONExample.json" in cwd
        PrintWriter pw = new PrintWriter("JSONExample.json");
        pw.write(jo.toJSONString());

        pw.flush();
        pw.close();
    }
}
```

Output from file “JSONExample.json” :

```
{
     "lastName":"Smith",
    "address":{
        "streetAddress":"21 2nd Street",
         "city":"New York",
         "state":"NY",
         "postalCode":10021
    },
     "age":25,
     "phoneNumbers":[
            {
            "type":"home", "number":"212 555-1234"
            },
         {
            "type":"fax", "number":"212 555-1234"
         }
     ],
     "firstName":"John"
}
```

**Note :** In JSON, An object is an unordered set of name/value pairs, so JSONObject doesn't preserve the order of an object's name/value pairs, since it is (by definition) not significant. Hence in our output file, order is not preserved.

## Read JSON from a file

Let us see an example that read JSON data from above created file "JSONExample.json" with help of JSONParser, JSONObject and JSONArray.

```
// Java program to read JSON from a file


import java.io.FileReader;

import java.util.Iterator;

import java.util.Map;


import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

import org.json.simple.parser.*;


public class JSONReadExample

{

    public static void main(String[] args) throws Exception

    {

        // parsing file "JSONExample.json"

        Object obj = new JSONParser().parse(new FileReader("JSONExample.json"));


        // typecasting obj to JSONObject

        JSONObject jo = (JSONObject) obj;


        // getting firstName and lastName

        String firstName = (String) jo.get("firstName");

        String lastName = (String) jo.get("lastName");


        System.out.println(firstName);
```

```
System.out.println(lastName);

// getting age
long age = (long) jo.get("age");
System.out.println(age);

// getting address
Map address = ((Map)jo.get("address"));

// iterating address Map
Iterator<Map.Entry> itr1 = address.entrySet().iterator();
while (itr1.hasNext()) {
    Map.Entry pair = itr1.next();
    System.out.println(pair.getKey() + " : " + pair.getValue());
}

// getting phoneNumbers
JSONArray ja = (JSONArray) jo.get("phoneNumbers");

// iterating phoneNumbers
Iterator itr2 = ja.iterator();

while (itr2.hasNext())
{
    itr1 = ((Map) itr2.next()).entrySet().iterator();
    while (itr1.hasNext()) {
        Map.Entry pair = itr1.next();
        System.out.println(pair.getKey() + " : " + pair.getValue());
    }
}
```

```
    }
 }
```

Output:

```
John
Smith
25
streetAddress : 21 2nd Street
postalCode : 10021
state : NY
city : New York
number : 212 555-1234
type : home
number : 212 555-1234
type : fax
```