XML document Processing in Java using XPath and XSLT

WHAT IS DOM?
WHAT IS SAX?

- Document Object Model (DOM) = provides a standard interface for working with an X
  document in a tree hierarchy. ML
- Simple API for XM L (SAX) = lets a program parse an XML sequentially, based on
  document
  an event handling model.
- Both represent APIs.

DOM PARSER

- The DOM Parser loads the complete XML content into a Tree structure.
- The iteration is done through the Node and NodeList to get the content of the XML.
- The XML sample for parsing using DOM parser:

```
<students>
   <student id="1">
      <firstName nickname="Aga">Agamemnon</firstName>
      <lastName>Dandanache</lastName>
      <location>Romania</location>
   </student>
   <student id="2">
      <firstName nickname="Ferro">Ferrero</firstName>
      <lastName>Rocher</lastName>
      <location>Italy</location>
   </student>
   <student id="3">
      <firstName nickname="Leana">Ileana</firstName>
      <lastName>Cosanzeana</lastName>
      <location>FairyTaleLand</location>
   </student>
</students>
```

DOMPARSER CODE (I)

```
package seweblab4; import
java.util.ArrayList; import
java.util.List;
import
javax.xml.parsers.Documen
tBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
public class DOMParser {
  public static void main(String[]
  args) throws Exception {
    //Get the DOM Builder
    Factory
    DocumentBuilderFactory
    factory =
    DocumentBuilderFactory.new
    Instance();
     //Get the DOM Builder
    DocumentBuilder builder =
    factory.newDocumentBuilder(
```

DOMPARSER CODE (II)

```
//Iterating through the nodes and extracting the data.
   NodeList nodeList = document.getDocumentElement().getChildNodes();
   for (int i =0; i <nodeList.getLength(); i++) {
//Wehave encountered a <student> tag.
Node node = nodeList.item(i);
if (node instanceof Element)
  {Student stu =new Student();
  stu.id =node.getAttributes().getNamedItem("id").getNodeValue();
  NodeList childNodes =node.getChildNodes();
  for (int j =0; j <childNodes.getLength(); j++) {
       Node cNode = childNodes.item(j);
       //Identifying the child tag of student encountered.
       if (cNode instanceof Element) {
            String content =cNode.getLastChild().getTextContent().trim(); switch
            (cNode.getNodeName()) {
                case "firstName": stu.firstName =content; break;
                case "lastName": stu.lastName =content; break; case
                "location": stu.location =content; break; }}
  }stuList.add(stu); }}for
(Student s : stuList) {
 System.out.println(s); }}
```

```
Agamemnon Dandanache(1)Romania
Ferrero Rocher(2)Italy
Ileana Cosanzeana(3)FairyTaleLand
```

Executing XPATH (I)

```
import java.io.IOException;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
import javax.xml.parsers.*;
import javax.xml.xpath.*;


public class XPathExample  {


    public static void main(String[]
    args) throws
    ParserConfigurationException,
    SAXException, IOException,
      XPathExpressionException  {
        DocumentBuilderFactory
        domFactory =
        DocumentBuilderFactory.new
        Instance();
        domFactory.setNamespaceAw
        are(true);
        DocumentBuilder builder =
        domFactory.newDocumentBui
        lder();
        Document doc =
        builder.parse("students.xml");
        XPathFactory factory = XPathFactory.newInstance();
        XPath xpath = factory.newXPath();
```

Executing XPATH (I)

```
        expr = xpath.compile("count(/students/student)");
        Double count = (Double) expr.evaluate(doc, XPathConstants.NUMBER);
        System.out.println("Students no = " + count);

        //Is there any person named Agamemnon?
        expr = xpath.compile("count(/students/student[firstName='Agamemnon']) > 0");
        Boolean res = (Boolean) expr.evaluate(doc, XPathConstants.BOOLEAN);
        System.out.println(res);


        //get the attribute value of firstName
        expr = xpath.compile("/students/student/firstName");
        result = expr.evaluate(doc, XPathConstants.NODESET);
        nodes = (NodeList) result;
        for (int i = 0; i < nodes.getLength(); i++) {
            Node nNode = nodes.item(i);
            String atrNick =
      nNode.getAttributes().getNamedItem("nickname").getNodeValue();
            System.out.println(atrNick);
        }
    }
}
```

Exercises

- **1.** <document>
    - <reference href="http://www.google.ro/"/>
  - </document>
- Select the value of the attribute "href".
- 2.
- <jobs>
    - <job priority="critical" name="Müll rausbringen" />
    - <job priority="low" name="Möbel säubern" />
    - <job priority="low" name="Teppich reinigen" />
    - <job priority="medium" name="Fenster putzen" />
    - <job priority="high" name="Pflanzen gießen" />
- </jobs>
- Count the number of the jobs with "low“ priority.

- 3.
- <persons>
    - <person firstName="Hans" lastName="Mustermann" age="28" />
    - <person firstName="Herbert" lastName="Möllemann" age="33" />
    - <person firstName="Peter" lastName="Meier" age="37" />
    - <person firstName="Ulrike" lastName="Albrecht" age="45" />
  - </persons>
- Select all the persons older dhant 35.
- 4.
- Select all the persons with last name starting with H.
- 5.
- Select all the persons with first name smaller than 5 letters.