**北京邮电大学软件学院**

**2016－2017学年第一学期实验报告**

**课程名称： XML& Web Service**

**项目名称： Using DOM & SAX Parser to Split XML File**

**项目完成人：**

**姓名：\_\_肖逸敏\_\_\_\_\_\_学号：\_2014211990\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**指导教师：\_\_\_\_\_\_吴国仕\_\_\_\_\_\_\_\_\_\_\_\_**

**日 期： 2016 年 12 月 2 日**

1. **实验目的**

掌握DOM和SAX解析XML文件的两种方法。

1. **实验内容**

In the directory, there IPO.XML, IBM\_COMP.XML and ABC\_COMP.XML, please do followings:

1. Using the DOM model in ECLIPSE environment, divide IPO.XML file into two XML files,(IBM\_COMP.XML and ABC\_COMP.XML) based on the attribute value of “comp\_name” .

Coding the project.

2)Using the SAX model to retrieve all the attribute information from IPO.XML and display them in the console.

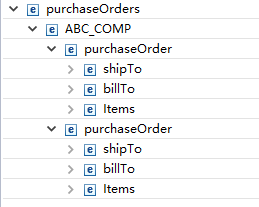
3）After finishing your development, write a word document to describe the different features of DOM, SAX with your development Experience.( Can use Chinese or English)

1. **实验环境**

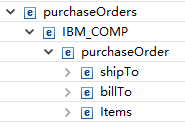
Eclipse，JAVA

1. **实验结果**

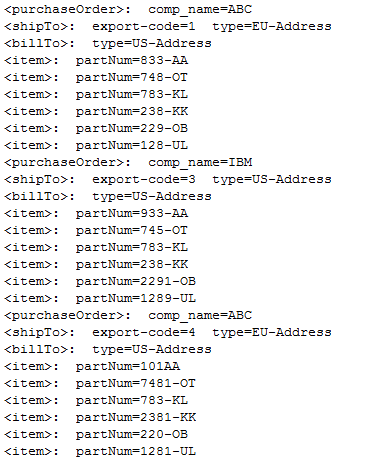
DOM生成的ABC\_COMP.xml:



生成的IBM\_COMP.xml:



SAX的生成结果：



1. **附录**

分析题目：

DOM方法的分析：

首先将源文件读取，将源文件有split分成两个部分，一个部分存放在doc\_ABC中，其中存放的是属性为comp\_name=”ABC”的节点以及子节点，另一部分存放在doc\_IBM中，其中存放的是属性为comp\_name=”IBM”的节点以及子节点。这两部分又分别用split\_ABC和split\_IBM两个函数进行拆分。

在split\_ABC中，还要区分是第一次碰到属性comp\_name=”ABC”还是第二次，因为第一次碰到后需要把节点<ABC\_COMP>插到当前节点的前面，而第二次的时候不需要。在split\_IBM中就不需要区分，相对而言要简单一些。

SAX方法的分析：

把所有的属性名和属性值输出，相较于DOM的方法要更简单，单个针对节点进行访问即可。

DOM和SAX方法的区分：

可以使用SAX来查询或者阅读XML文档。SAX可以快速扫描一个大型的XML文档，当它找到查询标准时就会立即停止，然后再处理之。DOM是把XML全部加载到内存中建立一棵树之后再进行处理。所以DOM不适合处理大型的XML。

同理，DOM的弱项就是SAX的强项，SAX不必把全部的xml都加载到内存中。但是SAX的缺点也很明显，它只能对文件顺序解析一遍，不支持对文件的随意存取。SAX也仅仅能够读取文件的内容，并不能修改内容。DOM可以随意修改文件树，从而修改了xml文件。

代码如下：

DOM代码：

package dom;

import java.io.File;

import javax.xml.parsers.\*;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import com.sun.org.apache.xerces.internal.dom.\*;

import org.w3c.dom.\*;

public class domSample {

private static Document doc;

private static Document doc\_ABC;

private static Document doc\_IBM;

private static File file;

private static File fileABC;

private static File fileIBM;

private static Element abcNode;

private static Element ibmNode;

public static void split(Node node){

if(node == null){

return;

}

int type = node.getNodeType();

switch(type){

case Node.DOCUMENT\_NODE:{

split(((Document)node).getDocumentElement());

break;

}

case Node.ELEMENT\_NODE:{

NodeList childNodes = node.getChildNodes();

int length = node.getAttributes().getLength();

if(length != 0){

//有属性

Attr attributes[] = new Attr[length];

for(int loopIndex = 0; loopIndex < length; loopIndex++){

attributes[loopIndex] =(Attr)node.getAttributes().item(loopIndex);

//属性有comp\_name="ABC"

if(attributes[loopIndex].getName().compareTo("comp\_name") ==0

&& attributes[loopIndex].getValue().compareTo("ABC") == 0){

abcNode.appendChild(doc\_ABC.importNode((Element)node, true));

}

//属性有comp\_name="IBM"

else if(attributes[loopIndex].getName().compareTo("comp\_name")==0

&& attributes[loopIndex].getValue().compareTo("IBM")==0){

ibmNode.appendChild(doc\_IBM.importNode((Element)node, true));

}

else

continue;

}

}

if (childNodes != null) {

for (int loopIndex = 0; loopIndex < childNodes.getLength(); loopIndex++ ) {

split(childNodes.item(loopIndex));

}

}

break;

}

case Node.COMMENT\_NODE:

break;

}

}

public static void split\_ABC(Node node,Element middle){

//Element middle=null;

Element newNode = null;

Element slibingNode = null;

if(node == null){

return;

}

int type = node.getNodeType();

switch(type){

case Node.ELEMENT\_NODE:{

NodeList childNodes = node.getChildNodes();

int length = node.getAttributes().getLength();

if(length != 0){

Attr attributes[] = new Attr[length];

for(int loopIndex = 0; loopIndex < length; loopIndex++){

attributes[loopIndex] =

(Attr)node.getAttributes().item(loopIndex);

}

for(int loopIndex = 0; loopIndex < length; loopIndex++){

//属性有comp\_name="ABC"

if(attributes[loopIndex].getName().compareTo("comp\_name") ==0

&& attributes[loopIndex].getValue().compareTo("ABC") == 0){

//ABC\_COMP.xml中没有ABC\_COMP结点

if(doc\_ABC.getElementsByTagName("ABC\_COMP").getLength() == 0){

newNode = doc\_ABC.createElement("ABC\_COMP");

((Element)node).removeAttributeNode(attributes[loopIndex]);

slibingNode = (Element)node.getNextSibling();

middle.insertBefore(newNode, node);

middle = newNode;

middle.appendChild(node);

middle.appendChild(slibingNode);

childNodes = middle.getChildNodes();

}

//ABC\_COMP.xml中有ABC\_COMP结点

else{

((Element)node).removeAttributeNode(attributes[loopIndex]);

middle =(Element) node;

}

}

else{

newNode = doc\_ABC.createElement(attributes[loopIndex].getName());

((Element)node).removeAttributeNode(attributes[loopIndex]);

newNode.appendChild(doc\_ABC.createTextNode(attributes[loopIndex].getValue()));

middle = (Element)node;

middle.appendChild(newNode);

}

}

}

else

middle = (Element)node;

if (childNodes != null) {

for (int loopIndex = 0; loopIndex < childNodes.getLength(); loopIndex++ ) {

split\_ABC(childNodes.item(loopIndex),middle);

}

}

break;

}

case Node.TEXT\_NODE:{

break;

}

}

}

public static void split\_IBM(Node node,Element middle){

Element newNode = null;

//Element middle=null;

if(node == null){

return;

}

int type = node.getNodeType();

switch(type){

case Node.ELEMENT\_NODE:{

NodeList childNodes = node.getChildNodes();

int length = node.getAttributes().getLength();

if(length != 0){

Attr attributes[] = new Attr[length];

for(int loopIndex = 0; loopIndex < length; loopIndex++){

attributes[loopIndex] =

(Attr)node.getAttributes().item(loopIndex);

}

for(int loopIndex = 0; loopIndex < length; loopIndex++){

//属性有comp\_name="IBM"

if(attributes[loopIndex].getName().compareTo("comp\_name") ==0

&& attributes[loopIndex].getValue().compareTo("IBM") == 0){

newNode = doc\_IBM.createElement("IBM\_COMP");

((Element)node).removeAttributeNode(attributes[loopIndex]);

middle.insertBefore(newNode, node);

middle = newNode;

middle.appendChild(node);

childNodes = middle.getChildNodes();

}

else{

newNode = doc\_IBM.createElement(attributes[loopIndex].getName());

((Element)node).removeAttributeNode(attributes[loopIndex]);

newNode.appendChild(doc\_IBM.createTextNode(attributes[loopIndex].getValue()));

middle = (Element)node;

middle.appendChild(newNode);

}

}

}

else

middle = (Element)node;

if (childNodes != null) {

for (int loopIndex = 0; loopIndex < childNodes.getLength(); loopIndex++ ) {

split\_IBM(childNodes.item(loopIndex),middle);

}

}

break;

}

case Node.TEXT\_NODE:{

break;

}

}

}

@SuppressWarnings("restriction")

public static void main(String args[]){

try{

try{

file = new File("src/ipo.xml");

fileABC = new File("src/ABC\_COMP.xml");

fileIBM = new File("src/IBM\_COMP.xml");

//获取解析器工厂

DocumentBuilderFactory factory =DocumentBuilderFactory.newInstance();

//获取解析器

DocumentBuilder builder\_ipo = factory.newDocumentBuilder();

//读取xml文档

doc = builder\_ipo.parse(file);

doc\_ABC = new DocumentImpl();

doc\_IBM = new DocumentImpl();

abcNode = doc\_ABC.createElement(doc.getDocumentElement().getNodeName());

doc\_ABC.appendChild(abcNode);

ibmNode = doc\_IBM.createElement(doc.getDocumentElement().getNodeName());

doc\_IBM.appendChild(ibmNode);

split(doc);

}

catch(Exception e){

e.printStackTrace(System.err);

}

split\_ABC(doc\_ABC.getDocumentElement(),null);

TransformerFactory transFactory = TransformerFactory.newInstance();

Transformer trans = transFactory.newTransformer();

DOMSource source1 = new DOMSource(doc\_ABC);

StreamResult result1 = new StreamResult(fileABC);

trans.transform(source1, result1);

split\_IBM(doc\_IBM.getDocumentElement(),null);

DOMSource source2 = new DOMSource(doc\_IBM);

StreamResult result2 = new StreamResult(fileIBM);

trans.transform(source2, result2);

System.out.println("done");

}

catch(TransformerException exc1){

exc1.printStackTrace();

}

}

}

SAX代码：

package sax;

import java.io.File;

import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

public class saxSample extends DefaultHandler{

String Tag=null ;

String nodeName=null;

public void startElement(String uri,String localName,String qName,Attributes attr)

throws SAXException{

if(attr.getLength() > 0){

System.out.print("<" + qName + ">: ");

for(int i = 0; i < attr.getLength();i++){

System.out.print(attr.getQName(i) + "=" + attr.getValue(i)+ " ");

}

System.out.print('\n');

}

Tag = qName;

}

public static void main(String args[])

{

saxSample split = new saxSample();

SAXParserFactory factory = SAXParserFactory.newInstance();

try{

SAXParser parser = factory.newSAXParser();

parser.parse(new File("src/ipo.xml"), split);

}

catch(Exception exc){

exc.printStackTrace();

}

}

}