**北京邮电大学软件学院**

**2017-2018 学年第 1学期实验报告**

**课程名称： XML&Web Service**

**实验名称： DOM&SAX\_\_\_\_\_ \_**

**实验完成人：**

**姓名：**\_\_\_\_原婷婷\_\_\_\_**学号：**\_\_\_\_\_2015212109\_\_\_

**成绩：**\_\_\_\_\_\_\_\_

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

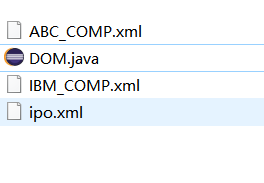
**日 期： 2017 年 12 月 27 日**

1. **实验目的**

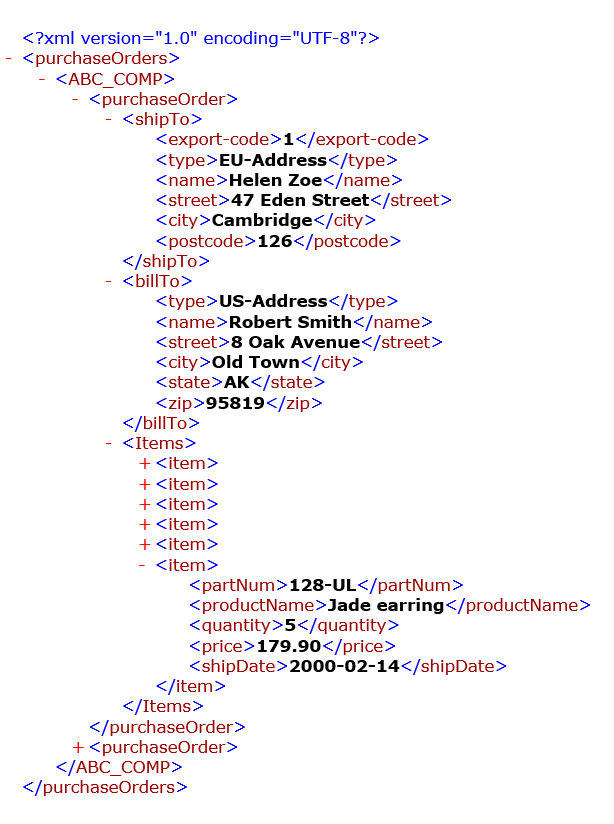
掌握DOM和SAX解析xml文件的方法。

1. **实验内容**
2. 使用DOM模型（eclipse环境）将ipo.xml文件按照不同公司名称分为两个文件：IBM\_COMP.XML，ABC\_COMP.XML。
3. 使用SAX模型从ipo.xml中检索所有属性信息，并在控制台中显示出来。
4. 描述DOM、SAX的不同功能以及开发体验。
5. **实验环境**
   1. Windows环境
   2. Java开发环境，eclipse java oxygen
6. **实验过程及结果**

DOM分离模型



ABC\_COMP和IBM\_COMP文档

SAX解析模型



1. **附录**
2. DOM&SAX

DOM模型：

读取源文件后，按照comp\_name属性分为两部分，扫描属性，分别调用list将两种存放在不同的文件中。已经列出了各节点，识别后分别创建其子节点，生成文件树。

SAX模型：

依次读入文件，直接将属性名和值显示在控制台。不需要存储，而且不用解析整个文档，较为简单，但是只能读取，不能修改。

区别：

SAX是基于事件的，是读入文件后产生相应的事件处理；DOM是基于树的，能对xml中数据进行更改，也能跨文件存取（拆分为两个文件），能持久保存在文件中。我认为，如果要对数据进行更改或者输出成xml文件，应当使用DOM，方便修改，如果文件比较大，只用于读取，用SAX比较合适。

1. 源代码

DOM.java

**public** **static** **void** main(String[] args){

*file\_ABC*=**new** File("src/xml/ABC\_COMP.xml");

*file\_IBM*=**new** File("src/xml/IBM\_COMP.xml");

DocumentBuilderFactory fac= DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = fac.newDocumentBuilder();

Document doc = builder.parse("ipo.xml");

NodeList listorder= doc.getElementsByTagName("purchaseOrder");

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

//当前的order

Element order= (Element) listorder.item(i);

String name= order.getAttributes().item(0).getNodeValue();

**if**(name.equals("ABC")){

*list*(listorder.item(i),*doc\_ABC*, *aCom*, *aOrder*,

*shipToA*, *billToA*, *itemsA*, *itemA*);

}

**else** **if**(name.equals("IBM")) {

*list*(listorder.item(i),*doc\_IBM*, *bCom*, *bOrder*,

*shipToB*, *billToB*, *itemsB*, *itemB*);

}

}

**private** **static** **void** list(Node node, Document doc, Element company, Element order, Element shipTo, Element billTo, Element items, Element item)

{

company.appendChild(order);

**for** (**int** i = 0; i < node.getChildNodes().getLength(); i++)

**if** (node.getChildNodes().item(i).getNodeType() == Node.***TEXT\_NODE***)

**continue**;

**else** **if** (node.getChildNodes().item(i).getNodeName().equals("shipTo")) {

order.appendChild(shipTo);

Element exportCode = doc.createElement("export-code");

v = node.getChildNodes().item(i).getAttributes().item(0).getNodeValue();

**if** (v == **null**)

**continue**;

value = doc.createTextNode(v);

shipTo.appendChild(exportCode);

exportCode.appendChild(value);

Element t = doc.createElement("type");

v = node.getChildNodes().item(i).getAttributes().item(1).getNodeValue();

**if** (v == **null**)

**continue**;

value = doc.createTextNode(v);

shipTo.appendChild(t);

t.appendChild(value);

listnode = node.getChildNodes().item(i).getChildNodes();

**for** (**int** j = 0; j < listnode.getLength(); j++)

**if** (listnode.item(j).getNodeType() != Node.***TEXT\_NODE***) {

Element c = doc.createElement(listnode.item(j).getNodeName().toString());

v = listnode.item(j).getTextContent();

**if** (v == **null**)

**continue**;

value = doc.createTextNode(v);

shipTo.appendChild(c);

c.appendChild(value);

}

}

**else** **if** (node.getChildNodes().item(i).getNodeName().equals("billTo")) {

order.appendChild(billTo);

Element t = doc.createElement("type");

v = node.getChildNodes().item(i).getAttributes().item(0).getNodeValue();

**if** (v == **null**)

**continue**;

value = doc.createTextNode(v);

t.appendChild(value);

billTo.appendChild(t);

listnode = node.getChildNodes().item(i).getChildNodes();

**for** (**int** j = 0; j < listnode.getLength(); j++)

**if** (listnode.item(j).getNodeType() != Node.***TEXT\_NODE***) {

Element c = doc.createElement(listnode.item(j)

.getNodeName().toString());

v = listnode.item(j).getTextContent();

**if** (v == **null**)

**continue**;

value = doc.createTextNode(v);

billTo.appendChild(c);

c.appendChild(value);

}

}

**else** **if** (node.getChildNodes().item(i).getNodeName().equals("Items")) {

order.appendChild(items);

NodeList itemList = node.getChildNodes().item(i).getChildNodes();

**for** (**int** j = 0; j < itemList.getLength(); j++)

**if** (itemList.item(j).getNodeType() == Node.***TEXT\_NODE***)

**continue**;

**else** {

item = doc.createElement("item");

items.appendChild(item);

Element patrNum = doc.createElement("partNum");

v = itemList.item(j).getAttributes().item(0).getNodeValue();

**if** (v == **null**)

**continue**;

value = doc.createTextNode(v);

item.appendChild(patrNum);

patrNum.appendChild(value);

listnode = itemList.item(j).getChildNodes();

**for** (**int** k = 0; k < listnode.getLength(); k++)

**if** (listnode.item(k).getNodeType() != Node.***TEXT\_NODE***) {

Element c = doc.createElement(listnode.item(k).getNodeName().toString());

v = listnode.item(k).getTextContent();

**if** (v == **null**)

**continue**;

value = doc.createTextNode(v);

item.appendChild(c);

c.appendChild(value);

}

}

}

}

SAX.java

**public** **class** SAX **extends** DefaultHandler{

String value=**null** ;

String node=**null**;

**public** **static** **void** main(String args[])

{

System.***out***.print("main.\n");

SAX ms = **new** SAX();

SAXParserFactory factory = SAXParserFactory.*newInstance*();

**try**{

SAXParser parser = factory.newSAXParser();

parser.parse(**new** File("ipo.xml"), ms);

}

**catch**(Exception exc){

exc.printStackTrace();

}

}

**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');

}

value = qName;

}