Java 核心技术(进阶)

第四章 高级文件处理

第二节 XML解析

华东师范大学 陈良育

XML解析

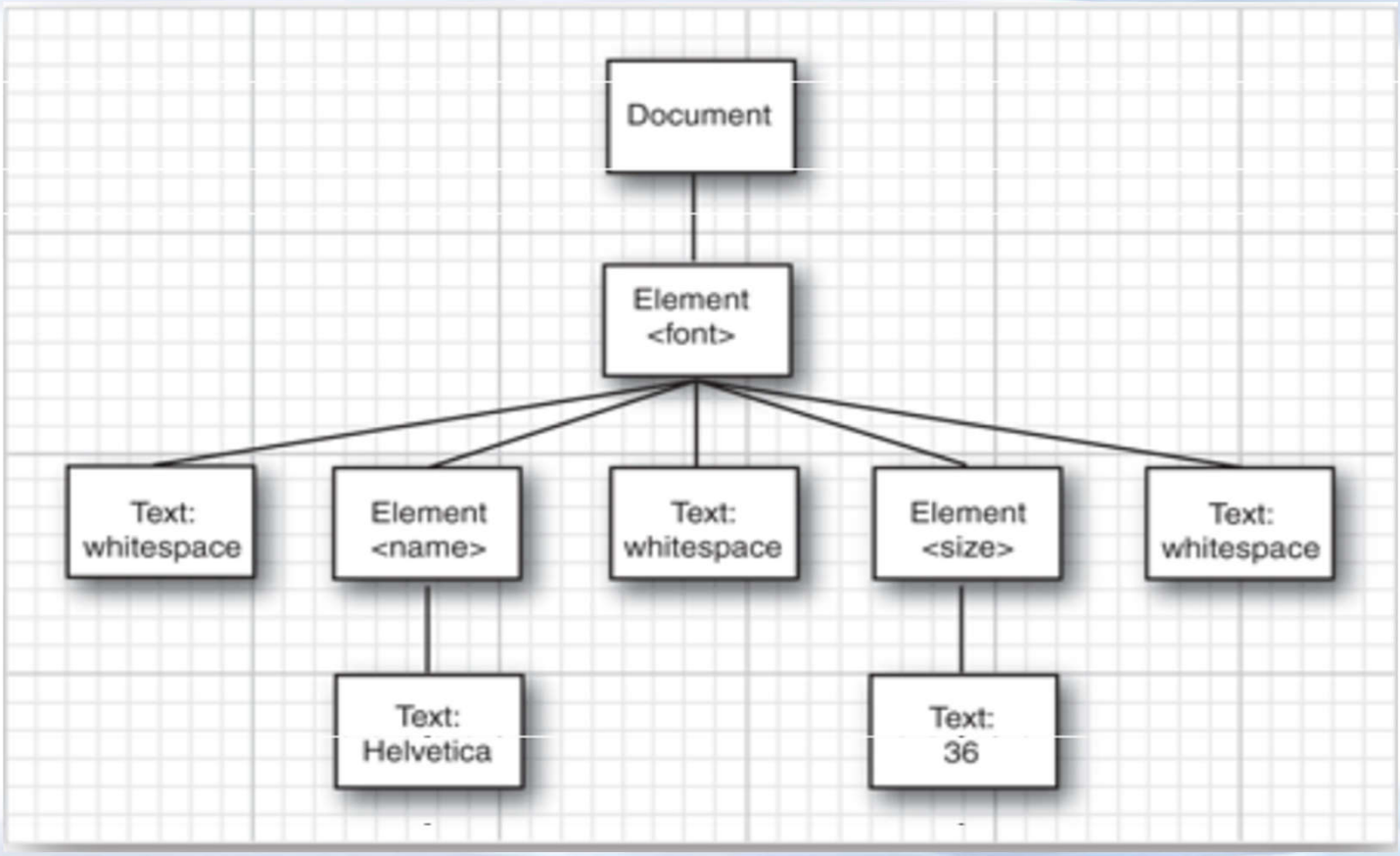
- XML解析方法
  - 树结构
    - DOM: Document Object Model 文档对象模型，擅长(小规模)读/写
  - 流结构
    - SAX: Simple API for XML 流机制解释器(推模式)，擅长读
    - Stax: The Streaming API for XML 流机制解释器(拉模式)，擅长读，JDK 6 引入

DOM

- DOM 是 W3C 处理 XML 的标准 API
  - 直观易用。
  - 其处理方式是将 XML 整个作为类似树结构的方式读入内存中以便操作及解析，方便修改。
  - 解析大数据量的 XML 文件，会遇到内存泄露及程序崩溃的风险。

DOM示例

```
<font>
    <name>Helvetica</name>
    <size>36</size>
</font>
```



DOM类

- DocumentBuilder 解析类，parse方法
- Node 节点主接口，getChildNodes返回一个NodeList
- NodeList 节点列表，每个元素是一个Node
- Document 文档根节点
- Element 标签节点元素 (每一个标签都是标签节点)
- Text节点 (包含在XML元素内的，都算Text节点)
- Attr节点(每个属性节点)
- 查看相关例子

SAX

- Simple API for XML
  - 采用事件/流模型来解析 XML 文档，更快速、更轻量。
  - 有选择的解析和访问，不像 DOM 加载整个文档，内存要求较低。
  - SAX 对 XML 文档的解析为一次性读取，不创建/不存储文档对象，很难同时访问文档中的多处数据。
  - 推模型。当它每发现一个节点就引发一个事件，而我们需要编写这些事件的处理程序。关键类：
- 查看相关例子

Stax

- Streaming API for XML
  - 流模型中的拉模型
  - 在遍历文档时，会把感兴趣的部分从读取器中拉出，不需要引发事件，允许我们选择性地处理节点。这大大提高了灵活性，以及整体效率。
  - 两套处理API
    - 基于指针的API， XMLStreamReader
    - 基于迭代器的API，XMLEventReader
- 查看相关例子

其他的第三方库

- DOM/SAX/Stax是JDK自带的解析功能。
- 第三方库
  - JDOM: www.jdom.org
  - DOM4J: dom4j.github.io
- 第三方库一般都包含DOM,SAX等多种方式解析，是对Java解析进行封装。

总结

- DOM：读(小规模)XML，写XML
- SAX/Stax：适合读(大规模)XML (一般大于1M，是并发量)

代码(1) DomReader.java


```
package xml.dom;

import javax.xml.parsers.DocumentBuilder;


public class DomReader
{
    public static void main(String[] a)
    {
        recursiveTraverse(); //自上而下进行访问
        System.out.println("=========华丽丽的分割线==========");
        traverseBySearch();     //根据名称进行搜索
    }
    public static void recursiveTraverse()
    {
        try
        {
            //采用Dom解析xml文件
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document document = db.parse("users.xml");

            //获取所有的一级子节点
            NodeList usersList = document.getChildNodes();
            System.out.println(usersList.getLength());  //1
```

代码(2) DomReader.java


```
for (int i = 0; i < usersList.getLength(); i++)
{
     Node users = usersList.item(i);           //1  users

     NodeList userList = users.getChildNodes(); //获取二级子节点user的列表
     System.out.println("==" + userList.getLength()); //9

     for (int j = 0; j < userList.getLength(); j++) //9
     {
          Node user = userList.item(j);
          if (user.getNodeType() == Node.ELEMENT_NODE)
          {
               NodeList metaList = user.getChildNodes();
               System.out.println("====" + metaList.getLength()); //7

               for (int k = 0; k < metaList.getLength(); k++) //7
               {
                   //到最后一级文本
                   Node meta = metaList.item(k);
                   if (meta.getNodeType() == Node.ELEMENT_NODE)
                   {
                        System.out.println(metaList.item(k).getNodeName()
                                + ":" + metaList.item(k).getTextContent());
                   }
               }
               System.out.println();
          }
     }
}
```

代码(3) DomReader.java

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void traverseBySearch()
    {
        try
        {
            //采用Dom解析xml文件
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document document = db.parse("users.xml");

            Element rootElement = document.getDocumentElement();

            NodeList nodeList = rootElement.getElementsByTagName("name");
            if(nodeList != null)
            {
               for (int i = 0 ; i < nodeList.getLength(); i++)
               {
                  Element element = (Element)nodeList.item(i);
                  System.out.println(element.getNodeName() + " = " + element.getTextContent());
               }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

代码(4) DomWriter.java


```
package xml.dom;

import java.io.File;

public class DomWriter {

    public static void main(String[] args) {


        try {
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();

            //新创建一个Document节点
            Document document = dbBuilder.newDocument();
            if (document != null)
            {
                Element docx = document.createElement("document");  //都是采用Document创建元素
                Element element = document.createElement("element");
                element.setAttribute("type", "paragraph");
                element.setAttribute("alignment", "left"); //element增加2个属性

                Element object = document.createElement("object");
                object.setAttribute("type", "text");
```

代码(5) DomWriter.java


```
            Element text = document.createElement("text");
            text.appendChild(document.createTextNode("abcdefg")); //给text节点赋值
            Element bold = document.createElement("bold");
            bold.appendChild(document.createTextNode("true"));     //给bold节点赋值

            object.appendChild(text);        //把text节点挂在object下
            object.appendChild(bold);        //把bold节点挂在object下
            element.appendChild(object);     //把object节点挂在element下
            docx.appendChild(element);       //把element节点挂在docx下
            document.appendChild(docx);      //把docx挂在document下

            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(document);

            //定义目标文件
            File file = new File("dom_result.xml");
            StreamResult result = new StreamResult(file);

            //将xml内容写入到文件中
            transformer.transform(source, result);

            System.out.println("write xml file successfully");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

代码(6) SAXReader.java


```
package xml.sax;

import java.io.IOException;

public class SAXReader {
    public static void main(String[] args) throws SAXException, IOException {
        XMLReader parser = XMLReaderFactory.createXMLReader();
        BookHandler bookHandler = new BookHandler();
        parser.setContentHandler(bookHandler);
        parser.parse("books.xml");
        System.out.println(bookHandler.getNameList());
    }
}
```

代码(7) SAXReader.java


```
class BookHandler extends DefaultHandler {
    private List<String> nameList;
    private boolean title = false;

    public List<String> getNameList() {
        return nameList;
    }

    // xml文档加载时
    public void startDocument() throws SAXException {
        System.out.println("Start parsing document...");
        nameList = new ArrayList<String>();
    }

    // 文档解析结束
    public void endDocument() throws SAXException {
        System.out.println("End");
    }
```

代码(8) SAXReader.java


```
    // 访问某一个元素
    public void startElement(String uri, String localName, String qName, Attributes atts) throws SAXException {

        if (qName.equals("title")) {
            title = true;
        }
    }

    // 结束访问元素
    public void endElement(String namespaceURI, String localName, String qName) throws SAXException {
        // End of processing current element
        if (title) {
            title = false;
        }
    }

    // 访问元素正文
    public void characters(char[] ch, int start, int length) {

        if (title) {
            String bookTitle = new String(ch, start, length);
            System.out.println("Book title: " + bookTitle);
            nameList.add(bookTitle);
        }
    }

}
```

代码(9) StaxReader.java


```
package xml.stax;

import java.io.FileNotFoundException;


public class StaxReader {

    public static void main(String[] args) {
        StaxReader.readByStream();
        System.out.println("=========华丽丽的分割线==========");
        StaxReader.readByEvent();
    }
```

代码(10) StaxReader.java


```
//流模式
public static void readByStream() {
    String xmlFile = "books.xml";
    XMLInputFactory factory = XMLInputFactory.newFactory();
    XMLStreamReader streamReader = null;
    try {
        streamReader = factory.createXMLStreamReader(new FileReader(xmlFile));
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (XMLStreamException e) {
        e.printStackTrace();
    }

    // 基于指针遍历
    try {
        while (streamReader.hasNext()) {
            int event = streamReader.next();
            // 如果是元素的开始
            if (event == XMLStreamConstants.START_ELEMENT) {
                // 列出所有书籍名称
                if ("title".equalsIgnoreCase(streamReader.getLocalName())) {
                    System.out.println("title:" + streamReader.getElementText());
                }
            }
        }
        streamReader.close();
    } catch (XMLStreamException e) {
        e.printStackTrace();
    }
}
```

代码(11) StaxReader.java


```
// 事件模式
public static void readByEvent() {
    String xmlFile = "books.xml";
    XMLInputFactory factory = XMLInputFactory.newInstance();
    boolean titleFlag = false;
    try {
        // 创建基于迭代器的事件读取器对象
        XMLEventReader eventReader = factory.createXMLEventReader(new FileReader(xmlFile));
        // 遍历Event迭代器
        while (eventReader.hasNext()) {
            XMLEvent event = eventReader.nextEvent();
            // 如果事件对象是元素的开始
            if (event.isStartElement()) {
                // 转换成开始元素事件对象
                StartElement start = event.asStartElement();
                // 打印元素标签的本地名称

                String name = start.getName().getLocalPart();
                //System.out.print(start.getName().getLocalPart());
                if(name.equals("title"))
                {
                    titleFlag = true;
                    System.out.print("title:");
                }
```

代码(12) StaxReader.java


```
        // 取得所有属性
        Iterator attrs = start.getAttributes();
        while (attrs.hasNext()) {
            // 打印所有属性信息
            Attribute attr = (Attribute) attrs.next();
            //System.out.print(":" + attr.getName().getLocalPart() + "=" + attr.getValue());
        }
        //System.out.println();
    }
    //如果是正文
    if(event.isCharacters())
    {
        String s = event.asCharacters().getData();
        if(null != s && s.trim().length()>0 && titleFlag)
        {
            System.out.println(s.trim());
        }
    }
```

代码(13) StaxReader.java


```
                //如果事件对象是元素的结束
                if(event.isEndElement())
                {
                    EndElement end = event.asEndElement();
                    String name = end.getName().getLocalPart();
                    if(name.equals("title"))
                    {
                        titleFlag = false;
                    }
                }
            }
            eventReader.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (XMLStreamException e) {
            e.printStackTrace();
        }
    }
}
```

谢 谢！