**说明：jdom解析XML文件方法**

SAXBuilder b = new SAXBuilder();  
Document doc = b.build(new File(xmlfilename));

//或

Document doc = b.build(new StringReader(Str));

2:一般而言，使用org.jdom.input.SAXBuilder更快，推荐使用，而org.jdom.input.DOMBuilder适用于已经

存在DOM对象的场合。

获取根元素：

doc.getRootElement();

取得子节点：

List allChildren = element.getChildren();//

List namedChildren = element.getChildren("name");

Element kid = element.**getChild**("name");//返回第一个

增加/删除子节点：

allChildren.remove(3); 移除指定位置  
allChildren.removeAll(element.getChildren("jack")); 移除指定元素

allChildren.add(new Element("jane")); 添加元素

allChildren.add(1, new Element("second")); 添加指定位置 元素

读取/设置属性：

String value =table.getAttributeValue("width");

//类型转换

try { value =table.getAttribute("border").getIntValue(); }

设置

table.addAttribute("vspace", "0");

table.addAttribute(new Attribute("name", "value"));  
table.removeAttribute("border");  
table.getAttributes().clear();

获取元素的内容:

String content = element.getText();

element.getTextNormalize();//移除多余的空白，字符串前后的空白，不会移除字符串内部的空白

设置元素的内容:

element.setText("description");

element.setText("<xml> content");//特殊字符可以被正确地转义

element.addContent(new CDATA("<xml> content"));//创建cdata元素

3 输出XML:

XMLOutputter 类用来实现XML文件的输出，在创建的时候需要一个Format对象来格式化XML文件，

Format对象是一个工厂类，提供几个静态的工厂方法来提供一些常规的XML格式，比如getPrettyFormat()

XMLOutputter outputter = new XMLOutputter(Format.getPrettyFormat());  
try { outputter.output(doc, new FileOutputStream(new File("xmlfile/persons.xml"))); }

//保存

Format format=Format.getCompactFormat();  
//设置文档字符编码  
format.setEncoding("UTF-8");  
//设置缩进字符串  
format.setIndent("  ");  
//XMLOutputter类提供了将JDOM树输出为字节流的能力  
XMLOutputter XMLOut=new XMLOutputter(format);  
//输出到文件  
XMLOut.output(doc,new FileOutputStream("student.xml"));

//输出到String

XMLOutputter output = new XMLOutputter();  
output.setFormat(Format.getPrettyFormat().setEncoding("UTF-8"));  
ByteArrayOutputStream bo = new ByteArrayOutputStream();

output.output(doc,bo);

xmlStr = bo.toString("UTF-8");

**Dom4j常用操作方法：**

//创建对象

Document document = DocumentHelper.createDocument();

//创建元素 添加元素的子元素

Element root = document.addElement("root");

//创建读取文件对象

SAXReader reader = new SAXReader();

// 通过read方法读取一个文件 转换成Document对象

reader.read(new File("student.xml"));  
// 获取根元素方法  
Element root = document.getRootElement();

//获取根元素的名称方法

root.getName(); // root students goods

//获取节点下的文本内容

root.getText();

//获取指定的子节点(label1)元素

root.element("label1");

//获取指定子节点(label1)的所有元素

root.elements("label1");

//获取root的name属性节点对象

Attribute attr = root.attribute("name");

//删除元素方法

Element.remove(titleElement);

4.设置某节点的属性和文字.

newMemberElm.addAttribute("name", "sitinspring");

5.设置属性的文字

             Attribute attribute=root.attribute("name");

             attribute.setText("sitinspring");

6.删除某属性

             Attribute attribute=root.attribute("size");//属性名name

             root.remove(attribute);

2.1、Document对象相关

1、读取XML文件,获得document对象.

        SAXReader reader = new SAXReader();

       Document   document = reader.read(new File("input.xml"));

2、解析XML形式的文本,得到document对象.

             String text = "<members></members>";

             Document document = DocumentHelper.parseText(text);

3、主动创建document对象.

             Document document = DocumentHelper.createDocument();

             Element root = document.addElement("members");// 创建根节点

2.2、节点相关

1、获取文档的根节点.

Element rootElm = document.getRootElement();

2、取得某节点的单个子节点.

Element memberElm=root.element("member"); //"member"是节点名

3.取得节点的文字

String text=memberElm.getText();

String text=root.elementText("name");这个是取得根节点下的name字节点的文字.

4.取得某节点下指定名称的所有节点并进行遍历.

List nodes = rootElm.elements("member");

for (Iterator it = nodes.iterator(); it.hasNext();) {

    Element elm = (Element) it.next();

   // do something

}

5.对某节点下的所有子节点进行遍历.

            for(Iterator it=root.elementIterator();it.hasNext();){

                 Element element = (Element) it.next();

             }

6.在某节点下添加子节点.

Element ageElm = new MemberElm.addElement("age");

7.设置节点文字.

ageElm.setText("29");

8.删除某节点.

parentElm.remove(childElm);    // childElm是待删除的节点,parentElm是其父节点

9.添加一个CDATA节点.

         Element contentElm = infoElm.addElement("content");

         contentElm.addCDATA(diary.getContent());

2.3、属性相关.

1.取得节点的指定的属性

             Element root=document.getRootElement();

             Attribute attribute=root.attribute("size");    // 属性名name

2.取得属性的文字

    String text=attribute.getText();

String text2=root.element("name").attributeValue("firstname");

//这个是取得根节点下name字节点的firstname属性的值.

3.遍历某节点的所有属性

Element root=document.getRootElement();

for(Iterator it=root.attributeIterator();it.hasNext();){

Attribute attribute = (Attribute) it.next();

String text=attribute.getText();

}

4.设置某节点的属性和文字.

newMemberElm.addAttribute("name", "sitinspring");

5.设置属性的文字

             Attribute attribute=root.attribute("name");

             attribute.setText("sitinspring");

6.删除某属性

             Attribute attribute=root.attribute("size");// 属性名name

             root.remove(attribute);

2.4、将文档写入XML文件.

1.文档中全为英文,不设置编码,直接写入.

XMLWriter writer = new XMLWriter(new FileWriter("output.xml"));

writer.write(document);

writer.close();

2.文档中含有中文,设置编码格式再写入.

OutputFormat format = OutputFormat.createPrettyPrint();

format.setEncoding("GBK");    // 指定XML编码

XMLWriter writer = new XMLWriter(new FileWriter("output.xml"),format);

writer.write(document);

writer.close();

2.5、字符串与XML的转换

1.将字符串转化为XML

String text = "<members> <member>sitinspring</member> </members>";

Document document = DocumentHelper.parseText(text);

2.将文档或节点的XML转化为字符串.

SAXReader reader = new SAXReader();

Document document = reader.read(new File("input.xml"));

Element root=document.getRootElement();

String docXmlText=document.asXML();

String rootXmlText=root.asXML();

Element memberElm=root.element("member");

String memberXmlText=memberElm.asXML();

3、dom4j的事件处理模型涉及的类和接口：

3.1、类：SAXReader

         public void addHandler(String path,ElementHandler handler)

     当解析到path指定的路径时，将调用参数handler指定的处理器。针对不同的节点可以添加多个handler实例。或者调用默认的Handler setDefaultHandler(ElementHandler handler);

3.2、接口ElementHandler

         public void onStart(ElementPath path)

     该方法在解析到元素的开始标签时被调用。

         public void onEnd(ElementPath path)

     该方法在解析到元素的结束标签时被调用

3.3、接口：ElementPath （假设有参数：ElementPath path）

         public void addHandler(String path,ElementHandler)

     该方法与SAXReader类中的addHandler()方法的作用相同。路径path可以是绝对路径（路径以/开头），也可以是相对路径（假设是当前路径的子节点路径）。

         public void removeHandler(String path)

     移除指定路径上的ElementHandler实例。路径可以是相对路径，也可以是绝对路径。

         public String getPath()

     该方法得到当前节点的路径。该方法返回的是完整的绝对路径

         public Element getCurrent()

     该方法得到当前节点。