dom4j api

笔记本: 从零开始造Spring
创建时间: 2018/6/24 11:50 **更新时间:** 2018/6/24 11:58
作者: 335131226@qq.com
URL: https://blog.csdn.net/u010781856/article/details/52601748

1、DOM4J简介

DOM4J是 dom4j.org 出品的一个开源 XML 解析包。DOM4J应用于 Java 平台，采用了 Java 集合框架并完全支持 DOM，SAX 和 JAXP。

DOM4J 使用起来非常简单。只要你了解基本的 XML-DOM 模型，就能使用。

Dom：把整个文档作为一个对象。

DOM4J 最大的特色是使用大量的接口。它的主要接口都在org.dom4j里面定义：

Attribute	定义了 XML 的属性。
Branch	指能够包含子节点的节点。如XML元素(Element)和文档(Docuemnts)定义了一个公共的行为
CDATA	定义了 XML CDATA 区域
CharacterData	是一个标识接口，标识基于字符的节点。如CDATA，Comment, Text.
Comment	定义了 XML 注释的行为
Document	定义了XML 文档
DocumentType	定义 XML DOCTYPE 声明
Element	定义XML 元素
ElementHandler	定义了Element 对象的处理器
ElementPath	被 ElementHandler 使用，用于取得当前正在处理的路径层次信息
Entity	定义 XML entity
Node	为dom4j中所有的XML节点定义了多态行为
NodeFilter	定义了在dom4j 节点中产生的一个滤镜或谓词的行为（predicate）
ProcessingInstruction	定义 XML 处理指令
Text	定义 XML 文本节点
Visitor	用于实现 Visitor模式
XPath	在分析一个字符串后会提供一个 XPath 表达式

interface java.lang.Cloneable接口之间的继承关系如下：

interface org.dom4j.**Node**

interface org.dom4j.Attribute

interface org.dom4j.Branch
interface org.dom4j.Document
interface org.dom4j.Element
interface org.dom4j.CharacterData
interface org.dom4j.CDATA
interface org.dom4j.Comment
interface org.dom4j.Text
interface org.dom4j.DocumentType
interface org.dom4j.Entity
interface org.dom4j.ProcessingInstruction

2、XML文档操作1

2.1、读取XML文档:

读写XML文档主要依赖于org.dom4j.io包，有DOMReader和SAXReader两种方式。因为利用了相同的接口，它们的调用方式是一样的。

```
public static Document load(String filename) {
Document document = null;
try {
SAXReader saxReader = new SAXReader();
document = saxReader.read(new File(filename)); //读取XML文件,获得document对象
} catch (Exception ex) {
ex.printStackTrace();
}
return document;
}
```

或

```
public static Document load(URL url) {
Document document = null;
try {
SAXReader saxReader = new SAXReader();
document = saxReader.read(url); //读取XML文件,获得document对象
} catch (Exception ex) {
ex.printStackTrace();
}
return document;
}
```

//读取指定的xml文件之后返回一个Document对象，这个对象代表了整个XML文档，用于各种Dom运算。执照XML文件头所定义的编码来转换。

2.2、获取根节点

根节点是xml分析的开始，任何xml分析工作都需要从根开始

```
Xml xml = new Xml();
Document dom = xml.load(path + "/" + file);
```

```
Element root = dom.getRootElement();
```

2.3、. 新增一个节点以及其下的子节点与数据

```
Element menuElement = root.addElement("menu");
Element engNameElement = menuElement.addElement("engName");
engNameElement.setText(catNameEn);
Element chiNameElement = menuElement.addElement("chiName");
chiNameElement.setText(catName);
```

2.4、 写入XML文件

注意文件操作的包装类**是乱码的根源**

```
public static boolean doc2XmlFile(Document document, String filename) {
boolean flag = true;
try {
XMLWriter writer = new XMLWriter( new OutputStreamWriter(new FileOutputStream(filename),"UTF-8"));
writer.write(document);
writer.close();
} catch (Exception ex) {
flag = false;
ex.printStackTrace();
}
System.out.println(flag);
return flag;
}
```

Dom4j通过XMLWriter将Document对象表示的XML树写入指定的文件，并使用OutputFormat格式对象指定写入的风格和编码方法。调用OutputFormat.createPrettyPrint()方法可以获得一个默认的pretty print风格的格式对象。对OutputFormat对象调用setEncoding()方法可以指定XML文件的编码方法。

```
public void writeTo(OutputStream out, String encoding) throws UnsupportedEncodingException, IOException {
    OutputFormat format = OutputFormat.createPrettyPrint();
    format.setEncoding("gb2312");
    XMLWriter writer = new XMLWriter(System.out,format);
    writer.write(doc);
    writer.flush();
    return;
}
```

2. 5、遍历xml节点

对Document对象调用getRootElement()方法可以返回代表根节点的Element对象。拥有了一个Element对象后，可以对该对象调用elementIterator()方法获得它的子节点的Element对象们的一个迭代器。使用(Element)iterator.next()方法遍历一个iterator并把每个取出的元素转化为Element类型。

```
public boolean isOnly(String catNameEn,HttpServletRequest request,String xml) {
boolean flag = true;
String path = request.getRealPath("");
```

```
Document doc = load(path+"/"+xml);
Element root = doc.getRootElement();
for (Iterator i = root.elementIterator(); i.hasNext();) {
Element el = (Element) i.next();
if(catNameEn.equals(el.elementTextTrim("engName"))){
flag = false;
break;
}
}
return flag;
}
```

2.6、创建xml文件

```
public static void main(String args[]){
String fileName="c:/text.xml";
Document document=DocumentHelper.createDocument();//建立document对象，用来操作xml文件
Element booksElement=document.addElement("books");//建立根节点
booksElement.addComment("This is a test for dom4j ");//加入一行注释
Element bookElement=booksElement.addElement("book");//添加一个book节点
bookElement.addAttribute("show","yes");//添加属性内容
Element titleElement=bookElement.addElement("title");//添加文本节点
titleElement.setText("ajax in action");//添加文本内容
try{
XMLWriter writer=new XMLWriter(new FileWriter(new File(fileName)));

writer.close();writer.write(document);

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

2.7、修改节点属性

```
public static void modifyXMLFile() {
String oldStr = "c:/text.xml";
String newStr = "c:/text1.xml";
Document document = null;
//修改节点的属性
try {
SAXReader saxReader = new SAXReader(); // 用来读取xml文档
document = saxReader.read(new File(oldStr)); // 读取xml文档
List list = document.selectNodes("/books/book/@show");// 用xpath查找节点book的属性
Iterator iter = list.iterator();
```

```
while (iter.hasNext()) {
Attribute attribute = (Attribute) iter.next();
if (attribute.getValue().equals("yes"))
    attribute.setValue("no");
}
} catch (Exception e) {
    e.printStackTrace();
}
//修改节点的内容
try {
SAXReader saxReader = new SAXReader(); // 用来读取xml文档
document = saxReader.read(new File(oldStr)); // 读取xml文档
List list = document.selectNodes("/books/book/title");// 用xpath查找节点book的内容
Iterator iter = list.iterator();
while (iter.hasNext()) {
Element element = (Element) iter.next();
element.setText("xxx");// 设置相应的内容
}
} catch (Exception e) {
    e.printStackTrace();
}

try {
XMLWriter writer = new XMLWriter(new FileWriter(new File(newStr)));
writer.write(document);
writer.close();
} catch (Exception ex) {
    ex.printStackTrace();
}
}
```

2.8、删除节点

```
public static void removeNode() {
String oldStr = "c:/text.xml";
String newStr = "c:/text1.xml";
Document document = null;
try {
SAXReader saxReader = new SAXReader();// 用来读取xml文档
document = saxReader.read(new File(oldStr));// 读取xml文档
List list = document.selectNodes("/books/book");// 用xpath查找对象
Iterator iter = list.iterator();
while (iter.hasNext()) {
```

```
Element bookElement = (Element) iter.next();
// 创建迭代器，用来查找要删除的节点,迭代器相当于指针，指向book下所有的title节点
Iterator iterator = bookElement.elementIterator("title");
while (iterator.hasNext()) {
Element titleElement = (Element) iterator.next();
if (titleElement.getText().equals("ajax in action")) {
bookElement.remove(titleElement);
}
}
}
} catch (Exception e) {
e.printStackTrace();
}
try {
XMLWriter writer = new XMLWriter(new FileWriter(new File(newStr)));
writer.write(document);
writer.close();
} catch (Exception ex) {
ex.printStackTrace();
}
}
```

2、XML文档操作2

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();
           // do something
         }
```

6.在某节点下添加子节点.

```
Element ageElm = newMemberElm.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();
System.out.println(text);
}
```

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。

3.3、Element类

getQName()	元素的QName对象
getNamespace()	元素所属的Namespace对象
getNamespacePrefix()	元素所属的Namespace对象的prefix
getNamespaceURI()	元素所属的Namespace对象的URI
getName()	元素的local name
getQualifiedName()	元素的qualified name
getText()	元素所含有的text内容，如果内容为空则返回一个空字符串而不是null
getTextTrim()	元素所含有的text内容，其中连续的空格被转化为单个空格，该方法不会返回null
attributeIterator()	元素属性的iterator，其中每个元素都是Attribute对象
attributeValue()	元素的某个指定属性所含的值
elementIterator()	元素的子元素的iterator，其中每个元素都是Element对象
element()	元素的某个指定（qualified name或者local name）的子元素
elementText()	元素的某个指定（qualified name或者local name）的子元素中的text信息
getParent	元素的父元素
getPath()	元素的XPath表达式，其中父元素的qualified name和子元素的qualified name之间使用"/"分隔
isTextOnly()	是否该元素只含有text或是空元素
isRootElement()	是否该元素是XML树的根节点