学习：**Dom4j**和**Xpath**

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：

魂牵梦萦

魂牵梦萦

魂牵梦萦

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(Documents)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

昵称：iloveJava
园龄：3年1个月
粉丝：104
关注：2
+加关注

< 2011年6月 >

日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

随笔分类

《Windows内核情景分析》(2)
Java工具(6)
Java设计模式(11)
Java异常处理(6)
Java原理学习(2)
MFC开发(23)
读书笔记(29)
面试工作(2)
驱动开发(23)
小工具集合(2)
学习：Android开发(17)

随笔档案

2011年10月 (2)
2011年9月 (8)
2011年8月 (26)
2011年7月 (41)
2011年6月 (39)

Windows驱动程序模型设计笔记

最新评论

1. Re:学习：短信的收发及在android模拟

XPath	在分析一个字符串后会提供一个 XPath 表达式

接口之间的继承关系如下：

interface java.lang.Cloneable

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();
```

器之间实践（2）
为什么这个代码区域点不开呢，点击+号没反应
--メ__僋翫
2. Re:学习：java设计模式—工厂模式
受教了
--AB西帝
3. Re:Java编程题90道
89题
13的高次尾数和3的高次尾数一样，反正是尾数，前面的可以忽略，3的N次幂是个数组循环a=[3,9,7,1]，298嘛，Mod(298,4)=2,尾数是a[2-1]=9
--メ__僋翫
4. Re:Java编程题90道
87题
丙都知道四位的车号刚好是一个整数的平方了还说没记住车号，他是找打的节奏吗？蜀黍的棍子不是吃素的。。。。
--メ__僋翫
5. Re:学习：Dom4j和Xpath
zhidao.baodu.com
--遗失在夏日里的春风

阅读排行榜

1. 学习：java设计模式—工厂模式(40567)
2. 学习：Android框架(20818)
3. 学习：Dom4j和Xpath(13933)
4. 学习：jxl导出excel(9420)
5. Struts2文件上传的大小限制问题(4811)

评论排行榜

1. 学习：java设计模式—工厂模式(4)
2. 学习：jxl导出excel(3)
3. pragma comment的使用 pragma预处理指令详解(2)
4. Java编程题90道(2)
5. 学习：java原理—反射机制(1)

推荐排行榜

1. 学习：java设计模式—工厂模式(6)
2. 学习：Android框架(3)
3. 《Windows核心编程》学习笔记（1）– DLL基础(隐式调用DLL)(3)
4. 学习：Dom4j和Xpath(2)
5. 学习：java设计模式—动态代理模式(2)

```
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, IO
Exception {
        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.write(document);
writer.close();
}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

3.4、类DocumentHelper

DocumentHelper 是用来生成生成 XML 文档的工厂类

4、通过xpath查找指定的节点

采用**xpath**查找需要引入**jaxen-xx-xx.jar**，否则会报**java.lang.NoClassDefFoundError: org/jaxen/JaxenException**异常。

```
List list=document.selectNodes("/books/book/@show");
```

4.1、 xpath语法

1、选取节点

XPath 使用路径表达式在 XML 文档中选取节点，节点是沿着路径或者 step 来选取的。

常见的路径表达式：

表达式	描述
nodename	选取当前节点的所有子节点
/	从根节点选取
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置
.	选取当前节点
..	选取当前节点的父节点
@	选取属性

实例

路径表达式	结果
bookstore	选取 bookstore 元素的所有子节点
/bookstore	选取根元素 bookstore
bookstore/book	选取bookstore 下名字为 book的所有子元素。
//book	选取所有 book 子元素，而不管它们在文档中的位置。
bookstore//book	选取bookstore 下名字为 book的所有后代元素，而不管它们位于 bookstore 之下的什么位置。
//@lang	选取所有名为 lang 的属性。

2、谓语（**Predicates**）

谓语用来查找某个特定的节点或者包含某个指定的值的节点。

谓语被嵌在方括号中。

实例

常见的谓语的一些路径表达式：

路径表达式	结果
/bookstore/book[1]	选取属于 bookstore 子元素的第一个 book 元素。
/bookstore/book[last()]	选取属于 bookstore 子元素的最后一个 book 元素。
/bookstore/book[last()-1]	选取属于 bookstore 子元素的倒数第二个 book 元素。
/bookstore/book[position()<3]	选取最前面的两个属于 bookstore 元素的子元素的 book 元素。
//title[@lang]	选取所有拥有名为 lang 的属性的 title 元素。

//title[@lang='eng']	选取所有 title 元素，要求这些元素拥有值为 eng 的 lang 属性。
/bookstore/book[price>35.00]	选取所有 bookstore 元素的 book 元素，要求book元素的子元素 price 元素的值须大于 35.00。
/bookstore/book[price>35.00]/title	选取所有 bookstore 元素中的 book 元素的 title 元素，要求book元素的子元素 price 元素的值须大于 35.00

3、选取未知节点

XPath 通配符可用来选取未知的 XML 元素。

通配符	描述
*	匹配任何元素节点
@*	匹配任何属性节点
node()	匹配任何类型的节点

实例

路径表达式	结果
/bookstore/*	选取 bookstore 元素的所有子节点
//*	选取文档中的所有元素
//title[@*]	选取所有带有属性的 title 元素。

4、选取若干路径

通过在路径表达式中使用"|"运算符，您可以选取若干个路径。

实例

路径表达式	结果
//book/title | //book/price	选取所有 book 元素的 title 和 price 元素。
//title | //price	选取所有文档中的 title 和 price 元素。
/bookstore/book/title|//price	选取所有属于 bookstore 元素的 book 元素的 title 元素，以及文档中所有的 price 元素。

5、XPath 轴

轴可定义某个相对于当前节点的节点集。

轴名称	结果
ancestor	选取当前节点的所有先辈（父、祖父等）
ancestor-or-self	选取当前节点的所有先辈（父、祖父等）以及当前节点本身
attribute	选取当前节点的所有属性
child	选取当前节点的所有子元素。
descendant	选取当前节点的所有后代元素（子、孙等）。
descendant-or-self	选取当前节点的所有后代元素（子、孙等）以及当前节点本身。

following	选取文档中当前节点的结束标签之后的所有节点。
namespace	选取当前节点的所有命名空间节点
parent	选取当前节点的父节点。
preceding	选取文档中当前节点的开始标签之前的所有节点。
preceding-sibling	选取当前节点之前的所有同级节点。
self	选取当前节点。

6、路径

- 位置路径表达式

位置路径可以是绝对的，也可以是相对的。

绝对路径起始于正斜杠(/)，而相对路径不会这样。在两种情况中，位置路径均包括一个或多个步，每个步均被斜杠分割：

- 绝对位置路径：
- 相对位置路径：

```
/step/step/...
step/step/...
```

每个步均根据当前节点集之中的节点来进行计算。

- 步（step）包括：

轴（axis）：定义所选节点与当前节点之间的树关系

节点测试（node-test）：识别某个轴内部的节点

零个或者更多谓语（predicate）：更深入地提炼所选的节点集

步的语法：轴名称**::**节点测试**[**谓语**]**

实例

例子	结果
child::book	选取所有属于当前节点的子元素的 book 节点
attribute::lang	选取当前节点的 lang 属性
child::*	选取当前节点的所有子元素
attribute::*	选取当前节点的所有属性
child::text()	选取当前节点的所有文本子节点
child::node()	选取当前节点的所有子节点
descendant::book	选取当前节点的所有 book 后代
ancestor::book	选择当前节点的所有 book 先辈
ancestor-or-self::book	选取当前节点的所有book先辈以及当前节点（假如此节点是book节点的话）
child::*/child::price	选取当前节点的所有 price 孙。

7、XPath 运算符

运算符	描述	实例	返回值
|	计算两个节点集	//book | //cd	返回所有带有 book 和 ck 元素的节点集

+	加法	6 + 4	10
-	减法	6 - 4	2
*	乘法	6 * 4	24
div	除法	8 div 4	2
=	等于	price=9.80	如果 price 是 9.80，则返回 true。 如果 price 是 9.90，则返回 fasle。
!=	不等于	price!=9.80	如果 price 是 9.90，则返回 true。 如果 price 是 9.80，则返回 fasle。
<	小于	price<9.80	如果 price 是 9.00，则返回 true。 如果 price 是 9.90，则返回 fasle。
<=	小于或等于	price<=9.80	如果 price 是 9.00，则返回 true。 如果 price 是 9.90，则返回 fasle。
>	大于	price>9.80	如果 price 是 9.90，则返回 true。 如果 price 是 9.80，则返回 fasle。
>=	大于或等于	price>=9.80	如果 price 是 9.90，则返回 true。 如果 price 是 9.70，则返回 fasle。
or	或	price=9.80 or price=9.70	如果 price 是 9.80，则返回 true。 如果 price 是 9.50，则返回 fasle。
and	与	price>9.00 and price<9.90	如果 price 是 9.80，则返回 true。 如果 price 是 8.50，则返回 fasle。
mod	计算除法的余数	5 mod 2	1

分类: Java工具

iloveJava
关注 - 2
粉丝 - 104
+加关注

2 0

(请您对文章做出评价)

« 上一篇：错误1：Apache可移植运行库
» 下一篇：学习：WordXML格式初步分析

posted @ 2011-06-09 16:28 iloveJava 阅读(13934) 评论(1) 编辑 收藏

评论

#1楼 2014-06-19 09:40 | 遗失在夏日里的春风

http://zhidao.baidu.comzhidao.baodu.com

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

博客园首页 博问 新闻 闪存 程序员招聘 知识库

I Love Lina~