dom4j讲义

讲师：陈伟俊

概述

- DOM4J是dom4j.org出品的一个开源XML解析包
- Dom4j是一个易用的、开源的库，用于XML，XPath和XSLT。它应用于Java平台，采用了Java集合框架并完全支持DOM，SAX

XML源示例1：emp.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ROWDATA>
<ROW>
  <EMPNO>7369</EMPNO>
  <ENAME>SMITH</ENAME>
  <HIREDATE>1980-12-17</HIREDATE>
  <SAL>900.00</SAL>
</ROW>
<ROW>
  <EMPNO>7499</EMPNO>
  <ENAME>ALLEN</ENAME>
  <HIREDATE>1981-2-20</HIREDATE>
  <SAL>1600.00</SAL>
</ROW>
</ROWDATA>
```

XML源示例2：config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<hibernate-mapping>
    <class name="com.news.po.Admin" table="ADMIN">
        <id name="id" type="java.lang.Integer">
            <generator class="increment" />
        </id>
        <property name="adminname" type="java.lang.String"/>
        <property name="adminpwd" type="java.lang.String"/>
    </class>
</hibernate-mapping>
```

解析示例1

```
//SAX对象
SAXReader reader = new SAXReader();
//输入流
InputStream is = TestParse1.class.getResourceAsStream("emp.xml");
//文档对象
Document document = reader.read(is);
//获得根节点
Element root = document.getRootElement();
System.out.println("根节点名称：" + root.getName());
//获得下一级所有子节点
List<Element> list1 = root.elements();
//遍历子节点
for (Element e1 : list1) {
	System.out.println(e1.getName() + ":");
	//获得下一级的子节点，继续遍历子节点
	List<Element> list2 = e1.elements();
	for (Element e2 : list2) {
		System.out.print("\t" + e2.getName() + ":");
		System.out.println(e2.getStringValue());
	}
}
is.close();
```

解析示例1输出结果

根节点名称：ROWDATA
ROW:
EMPNO:7369
ENAME:SMITH
HIREDATE:1980-12-17
SAL:900.00
ROW:
EMPNO:7499
ENAME:ALLEN
HIREDATE:1981-2-20
SAL:1600.00

解析示例2

```
//SAX对象
SAXReader reader = new SAXReader();
//输入流
InputStream is = TestParse2.class.getResourceAsStream("/com/icss/xml/config.xml");
//文档对象
Document document = reader.read(is);
//获得根节点
Element root = document.getRootElement();
System.out.println("根节点名称：" + root.getName());
//遍历子节点
List<Element> list1 = root.elements("class");
for (Element e1 : list1) {
	//遍历子节点
	List<Element> list2 = e1.elements("property");
	for (Element e2 : list2) {
		System.out.println("name=" + e2.attributeValue("name"));
	}
}
is.close();
```

解析示例2输出结果

根节点名称：hibernate-mapping

name=adminname

name=adminpwd

解析示例3：利用xpath

```
SAXReader reader = new SAXReader();

Document document = reader.read(TestParse3.class.getResource("config.xml"));

//通过xpath定位到某个叶子节点，返回这个节点的集合
List<Node> list =  document.selectNodes("//hibernate-mapping/class/property");

for (Node node : list) {
	System.out.println("TagName=" + node.getName());
	//通过xpath定位name属性，返回属性值
	System.out.println("name=" + node.valueOf("@name"));
}

System.out.println("-----------------------------------");

//通过xpath定位到某个叶子节点，如果存在多个同名节点仅返回第一个
Node node = document.selectSingleNode("//hibernate-mapping/class/property");
System.out.println("TagName=" + node.getName());
System.out.println("name=" + node.valueOf("@name"));
```

解析示例3输出结果

TagName=property
name=adminname
TagName=property
name=adminpwd

TagName=property
name=adminname

解析示例4：利用xpath

```
SAXReader reader = new SAXReader();
Document document =
    reader.read(TestParse4.class.getResourceAsStream("emp.xml"));

List<Node> list = document.selectNodes("//ROWDATA/ROW/ENAME");
for (Node node : list) {
    System.out.println(node.getName() + "=" + node.getStringValue());
}

Node node = document.selectSingleNode("//ROWDATA/ROW/EMPNO");
System.out.println(node.getName() + "=" + node.getStringValue());
```

解析示例4输出结果

ENAME=SMITH

ENAME=ALLEN

EMPNO=7369

写入XML文件示例

```
// 创建一个文档对象
Document document = DocumentHelper.createDocument();
// 加入根节点
Element root = document.addElement("hibernate-mapping");
// 加入子节点
Element e1 = root.addElement("class");
// 设置属性和属性值
e1.addAttribute("name", "com.Student");
e1.addAttribute("table", "student");
// 用方法链加入子节点和属性
Element e2 = e1.addElement("property").addAttribute("name", "sid")
.addAttribute("column", "s_id");
Element e3 = e1.addElement("property").addAttribute("name", "sname")
.addAttribute("column", "s_name");
//文件输出流
FileWriter fw = new FileWriter("e:\\student.xml");
//美化格式对象
OutputFormat format = OutputFormat.createPrettyPrint();
//带格式的输出流
XMLWriter xw = new XMLWriter(fw,format);
// 生成xml文件
xw.write(document);
// 清除缓冲区
xw.close();
```