**详解Java解析XML的四种方法**

XML现在已经成为一种通用的数据交换格式,平台的无关性使得很多场合都需要用到XML。本文将详细介绍用Java解析XML的四种方法。

XML现在已经成为一种通用的数据交换格式,它的平台无关性,语言无关性,系统无关性,给数据集成与交互带来了极大的方便。对于XML本身的语法知识与技术细节,需要阅读相关的技术文献,这里面包括的内容有DOM(Document Object Model),DTD(Document Type Definition),SAX(Simple API for XML),XSD(Xml Schema Definition),XSLT(Extensible Stylesheet Language Transformations),具体可参阅w3c官方网站文档http://www.w3.org获取更多信息。

XML在不同的语言里解析方式都是一样的,只不过实现的语法不同而已。基本的解析方式有两种,一种叫SAX，另一种叫DOM。SAX是基于事件流的解析,DOM是基于XML文档树结构的解析。假设我们XML的内容和结构如下: 

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <employees>  <employee>  <name>ddviplinux</name>  <sex>m</sex>  <age>30</age>  </employee>  </employees> |

本文使用JAVA语言来实现DOM与SAX的XML文档生成与解析。   
首先定义一个操作XML文档的接口XmlDocument 它定义了XML文档的建立与解析的接口。

|  |
| --- |
| package com.alisoft.facepay.framework.bean;  /\*\*  \*  \* @author hongliang.dinghl  \* 定义XML文档建立与解析的接口  \*/  public interface XmlDocument {  /\*\*  \* 建立XML文档  \* @param fileName 文件全路径名称  \*/  public void createXml(String fileName);  /\*\*  \* 解析XML文档  \* @param fileName 文件全路径名称  \*/  public void parserXml(String fileName);  } |

**1.DOM生成和解析XML文档**

为 XML 文档的已解析版本定义了一组接口。解析器读入整个文档，然后构建一个驻留内存的树结构，然后代码就可以使用 DOM 接口来操作这个树结构。优点：整个文档树在内存中，便于操作；支持删除、修改、重新排列等多种功能；缺点：将整个文档调入内存（包括无用的节点），浪费时间和空间；使用场合：一旦解析了文档还需多次访问这些数据；硬件资源充足（内存、CPU）。

package com.alisoft.facepay.framework.bean;   
import java.io.FileInputStream;   
import java.io.FileNotFoundException;   
import java.io.FileOutputStream;   
import java.io.IOException;   
import java.io.InputStream;   
import java.io.PrintWriter;   
import javax.xml.parsers.DocumentBuilder;   
import javax.xml.parsers.DocumentBuilderFactory;   
import javax.xml.parsers.ParserConfigurationException;   
import javax.xml.transform.OutputKeys;   
import javax.xml.transform.Transformer;   
import javax.xml.transform.TransformerConfigurationException;   
import javax.xml.transform.TransformerException;   
import javax.xml.transform.TransformerFactory;   
import javax.xml.transform.dom.DOMSource;   
import javax.xml.transform.stream.StreamResult;   
import org.w3c.dom.Document;   
import org.w3c.dom.Element;   
import org.w3c.dom.Node;   
import org.w3c.dom.NodeList;   
import org.xml.sax.SAXException;   
/\*\*   
\*   
\* @author hongliang.dinghl   
\* DOM生成与解析XML文档   
\*/   
public class DomDemo implements XmlDocument {   
private Document document;   
private String fileName;   
public void init() {   
try {   
DocumentBuilderFactory factory = DocumentBuilderFactory   
.newInstance();   
DocumentBuilder builder = factory.newDocumentBuilder();   
this.document = builder.newDocument();   
} catch (ParserConfigurationException e) {   
System.out.println(e.getMessage());   
}   
}   
public void createXml(String fileName) {   
Element root = this.document.createElement("employees");   
this.document.appendChild(root);   
Element employee = this.document.createElement("employee");   
Element name = this.document.createElement("name");   
name.appendChild(this.document.createTextNode("丁宏亮"));   
employee.appendChild(name);   
Element sex = this.document.createElement("sex");   
sex.appendChild(this.document.createTextNode("m"));   
employee.appendChild(sex);   
Element age = this.document.createElement("age");   
age.appendChild(this.document.createTextNode("30"));   
employee.appendChild(age);   
root.appendChild(employee);   
TransformerFactory tf = TransformerFactory.newInstance();   
try {   
Transformer transformer = tf.newTransformer();   
DOMSource source = new DOMSource(document);   
transformer.setOutputProperty(OutputKeys.ENCODING, "gb2312");   
transformer.setOutputProperty(OutputKeys.INDENT, "yes");   
PrintWriter pw = new PrintWriter(new FileOutputStream(fileName));   
StreamResult result = new StreamResult(pw);   
transformer.transform(source, result);   
System.out.println("生成XML文件成功!");   
} catch (TransformerConfigurationException e) {   
System.out.println(e.getMessage());   
} catch (IllegalArgumentException e) {   
System.out.println(e.getMessage());   
} catch (FileNotFoundException e) {   
System.out.println(e.getMessage());   
} catch (TransformerException e) {   
System.out.println(e.getMessage());   
}   
}   
public void parserXml(String fileName) {   
try {   
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();   
DocumentBuilder db = dbf.newDocumentBuilder();   
Document document = db.parse(fileName);   
NodeList employees = document.getChildNodes();   
for (int i = 0; i < employees.getLength(); i++) {   
Node employee = employees.item(i);   
NodeList employeeInfo = employee.getChildNodes();   
for (int j = 0; j < employeeInfo.getLength(); j++) {   
Node node = employeeInfo.item(j);   
NodeList employeeMeta = node.getChildNodes();   
for (int k = 0; k < employeeMeta.getLength(); k++) {   
System.out.println(employeeMeta.item(k).getNodeName()   
+ ":" + employeeMeta.item(k).getTextContent());   
}   
}   
}   
System.out.println("解析完毕");   
} catch (FileNotFoundException e) {   
System.out.println(e.getMessage());   
} catch (ParserConfigurationException e) {   
System.out.println(e.getMessage());   
} catch (SAXException e) {   
System.out.println(e.getMessage());   
} catch (IOException e) {   
System.out.println(e.getMessage());   
}   
}   
}

**2.SAX生成和解析XML文档**

为解决DOM的问题，出现了SAX。SAX ，事件驱动。当解析器发现元素开始、元素结束、文本、文档的开始或结束等时，发送事件，程序员编写响应这些事件的代码，保存数据。优点：不用事先调入整个文档，占用资源少；SAX解析器代码比DOM解析器代码小，适于Applet，下载。缺点：不是持久的；事件过后，若没保存数据，那么数据就丢了；无状态性；从事件中只能得到文本，但不知该文本属于哪个元素；使用场合：Applet;只需XML文档的少量内容，很少回头访问；机器内存少；

Java代码

package com.alisoft.facepay.framework.bean;     
import java.io.FileInputStream;     
import java.io.FileNotFoundException;     
import java.io.IOException;     
import java.io.InputStream;     
  
import javax.xml.parsers.ParserConfigurationException;     
import javax.xml.parsers.SAXParser;     
import javax.xml.parsers.SAXParserFactory;     
  
import org.xml.sax.Attributes;     
import org.xml.sax.SAXException;     
import org.xml.sax.helpers.DefaultHandler;     
/\*\*    
\*     
\* @author hongliang.dinghl    
\* SAX文档解析    
\*/    
public class SaxDemo implements XmlDocument {     
  
public void createXml(String fileName) {     
System.out.println("<<"+filename+">>");     
}     
  
public void parserXml(String fileName) {     
SAXParserFactory saxfac = SAXParserFactory.newInstance();     
  
try {     
  
SAXParser saxparser = saxfac.newSAXParser();     
  
InputStream is = new FileInputStream(fileName);     
  
saxparser.parse(is, new MySAXHandler());     
  
} catch (ParserConfigurationException e) {     
  
e.printStackTrace();     
  
} catch (SAXException e) {     
  
e.printStackTrace();     
  
} catch (FileNotFoundException e) {     
  
e.printStackTrace();     
  
} catch (IOException e) {     
  
e.printStackTrace();     
  
}     
  
}     
  
}     
  
class MySAXHandler extends DefaultHandler {     
  
boolean hasAttribute = false;     
  
Attributes attributes = null;     
  
public void startDocument() throws SAXException {     
  
System.out.println("文档开始打印了");     
  
}     
  
public void endDocument() throws SAXException {     
  
System.out.println("文档打印结束了");     
  
}     
  
public void startElement(String uri, String localName, String qName,     
  
Attributes attributes) throws SAXException {     
  
if (qName.equals("employees")) {     
  
return;     
  
}     
  
if (qName.equals("employee")) {     
  
System.out.println(qName);     
  
}     
  
if (attributes.getLength() > 0) {     
  
this.attributes = attributes;     
  
this.hasAttribute = true;     
  
}     
  
}     
  
public void endElement(String uri, String localName, String qName)     
  
throws SAXException {     
  
if (hasAttribute && (attributes != null)) {     
  
for (int i = 0; i < attributes.getLength(); i++) {     
  
System.out.println(attributes.getQName(0)     
+ attributes.getValue(0));     
  
}     
  
}     
  
}     
  
public void characters(char[] ch, int start, int length)     
  
throws SAXException {     
  
System.out.println(new String(ch, start, length));     
  
}     
  
}    
package com.alisoft.facepay.framework.bean;   
import java.io.FileInputStream;   
import java.io.FileNotFoundException;   
import java.io.IOException;   
import java.io.InputStream;   
import javax.xml.parsers.ParserConfigurationException;   
import javax.xml.parsers.SAXParser;   
import javax.xml.parsers.SAXParserFactory;   
import org.xml.sax.Attributes;   
import org.xml.sax.SAXException;   
import org.xml.sax.helpers.DefaultHandler;   
/\*\*   
\*   
\* @author hongliang.dinghl   
\* SAX文档解析   
\*/   
public class SaxDemo implements XmlDocument {   
public void createXml(String fileName) {   
System.out.println("<<"+filename+">>");   
}   
public void parserXml(String fileName) {   
SAXParserFactory saxfac = SAXParserFactory.newInstance();   
try {   
SAXParser saxparser = saxfac.newSAXParser();   
InputStream is = new FileInputStream(fileName);   
saxparser.parse(is, new MySAXHandler());   
} catch (ParserConfigurationException e) {   
e.printStackTrace();   
} catch (SAXException e) {   
e.printStackTrace();   
} catch (FileNotFoundException e) {   
e.printStackTrace();   
} catch (IOException e) {   
e.printStackTrace();   
}   
}   
}   
class MySAXHandler extends DefaultHandler {   
boolean hasAttribute = false;   
Attributes attributes = null;   
public void startDocument() throws SAXException {   
System.out.println("文档开始打印了");   
}   
public void endDocument() throws SAXException {   
System.out.println("文档打印结束了");   
}   
public void startElement(String uri, String localName, String qName,   
Attributes attributes) throws SAXException {   
if (qName.equals("employees")) {   
return;   
}   
if (qName.equals("employee")) {   
System.out.println(qName);   
}   
if (attributes.getLength() > 0) {   
this.attributes = attributes;   
this.hasAttribute = true;   
}   
}   
public void endElement(String uri, String localName, String qName)   
throws SAXException {   
if (hasAttribute && (attributes != null)) {   
for (int i = 0; i < attributes.getLength(); i++) {   
System.out.println(attributes.getQName(0)   
+ attributes.getValue(0));   
}   
}   
}   
public void characters(char[] ch, int start, int length)   
throws SAXException {   
System.out.println(new String(ch, start, length));   
}   
}

**3.DOM4J生成和解析XML文档**

DOM4J 是一个非常非常优秀的Java XML API，具有性能优异、功能强大和极端易用使用的特点，同时它也是一个开放源代码的软件。如今你可以看到越来越多的 Java 软件都在使用 DOM4J 来读写 XML，特别值得一提的是连 Sun 的 JAXM 也在用 DOM4J。

Java代码

package com.alisoft.facepay.framework.bean;     
import java.io.File;     
import java.io.FileWriter;     
import java.io.IOException;     
import java.io.Writer;     
import java.util.Iterator;     
  
import org.dom4j.Document;     
import org.dom4j.DocumentException;     
import org.dom4j.DocumentHelper;     
import org.dom4j.Element;     
import org.dom4j.io.SAXReader;     
import org.dom4j.io.XMLWriter;     
/\*\*    
\*     
\* @author hongliang.dinghl    
\* Dom4j 生成XML文档与解析XML文档    
\*/    
public class Dom4jDemo implements XmlDocument {     
  
public void createXml(String fileName) {     
Document document = DocumentHelper.createDocument();     
Element employees=document.addElement("employees");     
Element employee=employees.addElement("employee");     
Element name= employee.addElement("name");     
name.setText("ddvip");     
Element sex=employee.addElement("sex");     
sex.setText("m");     
Element age=employee.addElement("age");     
age.setText("29");     
try {     
Writer fileWriter=new FileWriter(fileName);     
XMLWriter xmlWriter=new XMLWriter(fileWriter);     
xmlWriter.write(document);     
xmlWriter.close();     
} catch (IOException e) {     
  
System.out.println(e.getMessage());     
}     
  
  
}     
  
  
public void parserXml(String fileName) {     
File inputXml=new File(fileName);     
SAXReader saxReader = new SAXReader();     
try {     
Document document = saxReader.read(inputXml);     
Element employees=document.getRootElement();     
for(Iterator i = employees.elementIterator(); i.hasNext();){     
Element employee = (Element) i.next();     
for(Iterator j = employee.elementIterator(); j.hasNext();){     
Element node=(Element) j.next();     
System.out.println(node.getName()+":"+node.getText());     
}     
  
}     
} catch (DocumentException e) {     
System.out.println(e.getMessage());     
}     
System.out.println("dom4j parserXml");     
}     
}

**4.JDOM生成和解析XML**

为减少DOM、SAX的编码量，出现了JDOM；优点：20-80原则，极大减少了代码量。使用场合：要实现的功能简单，如解析、创建等，但在底层，JDOM还是使用SAX（最常用）、DOM、Xanan文档。

package com.alisoft.facepay.framework.bean;     
  
import java.io.FileNotFoundException;     
import java.io.FileOutputStream;     
import java.io.IOException;     
import java.util.List;     
  
import org.jdom.Document;     
import org.jdom.Element;     
import org.jdom.JDOMException;     
import org.jdom.input.SAXBuilder;     
import org.jdom.output.XMLOutputter;     
/\*\*    
\*     
\* @author hongliang.dinghl    
\* JDOM 生成与解析XML文档    
\*     
\*/    
public class JDomDemo implements XmlDocument {     
  
public void createXml(String fileName) {     
Document document;     
Element  root;     
root=new Element("employees");     
document=new Document(root);     
Element employee=new Element("employee");     
root.addContent(employee);     
Element name=new Element("name");     
name.setText("ddvip");     
employee.addContent(name);     
Element sex=new Element("sex");     
sex.setText("m");     
employee.addContent(sex);     
Element age=new Element("age");     
age.setText("23");     
employee.addContent(age);     
XMLOutputter XMLOut = new XMLOutputter();     
try {     
XMLOut.output(document, new FileOutputStream(fileName));     
} catch (FileNotFoundException e) {     
e.printStackTrace();     
} catch (IOException e) {     
e.printStackTrace();     
}     
  
}     
  
public void parserXml(String fileName) {     
SAXBuilder builder=new SAXBuilder(false);      
try {     
Document document=builder.build(fileName);     
Element employees=document.getRootElement();      
List employeeList=employees.getChildren("employee");     
for(int i=0;i  
Element employee=(Element)employeeList.get(i);     
List employeeInfo=employee.getChildren();     
for(int j=0;j  
System.out.println(((Element)employeeInfo.get(j)).getName()+":"+((Element)employeeInfo.get(j)).getValue());     
  
}     
}     
} catch (JDOMException e) {     
  
e.printStackTrace();     
} catch (IOException e) {     
  
e.printStackTrace();     
}      
  
}     
}  