java中四种操作（sax、dom、jdom、dom4j）xml方式详解与比较

**博客分类：** [xml](http://634082093-qq-com.iteye.com/category/259972)

网上能够查到很多的4种解析方式的区别，我再做一下摘录和总结，顺带给自己做个备份。

SAX

sax分析器在对xml文档进行分析时,触发一系列的事件,应用程序通过事件处理函数实现对xml文档的访问.

因为事件触发是有时序性的,所以sax分析器提供的是一种对xml文档的顺序访问机制,对于已经分析过的部分,

不能再重新倒回去处理.此外，它也不能同时访问处理2个tag.

sax分析器在实现时,只是顺序地检查xml文档中的字节流,判断当前字节是xml语法中的哪一部分,检查是否符合

xml语法并且触发相应的事件.对于事件处理函数的本身,要由应用程序自己来实现. SAX解析器采用了基于事件的模型，它在解析XML文档的时候可以触发一系列的事件，当发现给定的tag的时候，它可以激活一个回调方法，告诉该方法制定的标签已经找到。

同DOM分析器相比,sax分析器对xml文档的处理缺乏灵活性,但对于只访问xml文档中的数据而不改变应用程序

来说,sax分析器的效率会很高.

sax对内存要求也比较低.

优点:1、无需将整个文档加载到内存中,所以内存消耗少.

       2 、推模型允许注册多个contentHandler.

缺点:1、没有内置的文档导航支持.

       2、不能随机访问xml文档.

       3、不支持原地修改xml.

       4、不支持名字空间作用域.

适合：对xml文档读取数据

DOM

DOM解析器把XML文档转化为一个包含其内容的树，并可以对树进行遍历。也就是说，通过DOM树，应用程序可以对XML文档进行随机访问。这种访问方式给应用程序的开发带来了很大的灵活性，它可以任意地控制整个XML文档中的内容.用DOM解析模型编程容易，开发人员只需要调用建树的指令，然后利用navigation APIs访问所需的树节点来完成任务。可以很容易的添加和修改树中的元素。然而由于使用DOM解析器的时候需要处理整个XML文档，所以对性能和内存的要求比较高，尤其是遇到很大的XML文件的时候。由于它的遍历能力，DOM解析器常用于XML文档需要频繁的改变的服务中。

优点:1、丰富的API集合,可以轻松导航.

        2、 整个DOM树加载到内存,允许随机访问.

缺点:1、整个xml文档必须一次解析完.

        2、整个DOM树都加载到内存中,内存要求高.

        3、一般的DOM节点对于必须为所有节点创建对象的对象类型绑定不太理想.

适用：对xml文档修改数据

JDOM

JDOM是处理xml的纯java api.使用具体类而不是接口.JDOM具有树的遍历,又有SAX的java规则.JDOM与DOM主要有两方面不同。首先，JDOM仅使用具体类而不使用接口。这在某些方面简化了API，但是也限制了灵活性。第二，API大量使用了Collections类，简化了那些已经熟悉这些类的Java开发者的使用。

JDOM自身不包含解析器。它通常使用SAX2解析器来解析和验证输入XML文档（尽管它还可以将以前构造的DOM表示作为输入）。它包含一些转换器以将JDOM表示输出成SAX2事件流、DOM模型或XML文本文档。JDOM是在Apache许可证变体下发布的开放源码。

优点:1、是基于树的处理xml的java api,把树加载到内存中.

        2、没有向下兼容的限制,所以比DOM简单.

        3、速度快.

        4、具有SAX的java 规则.

缺点:1、不能处理大于内存的文档.

        2、JDOM表示XML文档逻辑模型,不能保证每个字节真正变换.

        3、 针对实例文档不提供DTD与模式的任何实际模型.

        4、 不支持于DOM中相应遍历包.

适用：自行选择

DOM4J

DOM4J有更复杂的api,所以dom4j比jdom有更大的灵活性.DOM4J性能最好，连Sun的JAXM也在用DOM4J.目前许多开源项目中大量采用DOM4J，例如大名鼎鼎的Hibernate也用DOM4J来读取XML配置文件。如果不考虑可移植性，那就采用DOM4J.

优点：1、灵活性最高

     2、易用性和功能强大、性能优异

缺点：1、复杂的api

     2、移植性差

适用：自行选择

比较

1）DOM4J性能最好，连Sun的JAXM也在用DOM4J.目前许多开源项目中大量采用DOM4J，例如大名鼎鼎的Hibernate也用DOM4J来读取XML配置文件。如果不考虑可移植性，那就采用DOM4J.

2）JDOM和DOM在性能测试时表现不佳，在测试10M文档时内存溢出。在小文档情况下还值得考虑使用DOM和JDOM.虽然JDOM的开发者已经说明他们期望在正式发行版前专注性能问题，但是从性能观点来看，它确实没有值得推荐之处。另外，DOM仍是一个非常好的选择。DOM实现广泛应用于多种编程语言。它还是许多其它与XML相关的标准的基础，因为它正式获得W3C推荐（与基于非标准的Java模型相对），所以在某些类型的项目中可能也需要它（如在JavaScript中使用DOM）。

3）SAX表现较好，这要依赖于它特定的解析方式－事件驱动。一个SAX检测即将到来的XML流，但并没有载入到内存（当然当XML流被读入时，会有部分文档暂时隐藏在内存中）。

四种xml操作方式的基本使用方法

xml文件：

＜?xml version="1.0" encoding="GB2312"?＞

＜RESULT＞

＜VALUE＞

　　 ＜NO＞A1234＜/NO＞

　　 ＜ADDR＞XX号＜/ADDR＞

＜/VALUE＞

＜VALUE＞

　　 ＜NO＞B1234＜/NO＞

　 　＜ADDR＞XX组＜/ADDR＞

＜/VALUE＞

＜/RESULT＞

1）DOM

import java.io.\*;

import java.util.\*;

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

public class MyXMLReader{

　public static void main(String arge[]){

　　long lasting =System.currentTimeMillis();

　　try{

　　　File f=new File("data\_10k.xml");

　　　DocumentBuilderFactory factory=DocumentBuilderFactory.newInstance();

　　　DocumentBuilder builder=factory.newDocumentBuilder();

　　　Document doc = builder.parse(f);

　　　NodeList nl = doc.getElementsByTagName("VALUE");

　　　for (int i=0;i＜nl.getLength();i++){

　　　　System.out.print("车牌号码:" + doc.getElementsByTagName("NO").item(i).getFirstChild().getNodeValue());

　　　　System.out.println("车主地址:" + doc.getElementsByTagName("ADDR").item(i).getFirstChild().getNodeValue());

　 　}

　　}catch(Exception e){

　　　e.printStackTrace();

}

2）SAX

import org.xml.sax.\*;

import org.xml.sax.helpers.\*;

import javax.xml.parsers.\*;

public class MyXMLReader extends DefaultHandler {

　java.util.Stack tags = new java.util.Stack();

　public MyXMLReader() {

　　super();

}

　public static void main(String args[]) {

　　long lasting = System.currentTimeMillis();

　　try {

　　　SAXParserFactory sf = SAXParserFactory.newInstance();

　　　SAXParser sp = sf.newSAXParser();

　　　MyXMLReader reader = new MyXMLReader();

　　　sp.parse(new InputSource("data\_10k.xml"), reader);

　　} catch (Exception e) {

　　　e.printStackTrace();

　　}

　　System.out.println("运行时间：" + (System.currentTimeMillis() - lasting) + "毫秒");}

　　public void characters(char ch[], int start, int length) throws SAXException {

　　String tag = (String) tags.peek();

　　if (tag.equals("NO")) {

　　　System.out.print("车牌号码：" + new String(ch, start, length));

}

if (tag.equals("ADDR")) {

　　System.out.println("地址:" + new String(ch, start, length));

}

}

　　public void startElement(String uri,String localName,String qName,Attributes attrs) {

　　tags.push(qName);}

}

3） JDOM

import java.io.\*;

import java.util.\*;

import org.jdom.\*;

import org.jdom.input.\*;

public class MyXMLReader {

　public static void main(String arge[]) {

　　long lasting = System.currentTimeMillis();

　　try {

　　　SAXBuilder builder = new SAXBuilder();

　　　Document doc = builder.build(new File("data\_10k.xml"));

　　　Element foo = doc.getRootElement();

　　　List allChildren = foo.getChildren();

　　　for(int i=0;i＜allChildren.size();i++) {

　　　　System.out.print("车牌号码:" + ((Element)allChildren.get(i)).getChild("NO").getText());

　　　　System.out.println("车主地址:" + ((Element)allChildren.get(i)).getChild("ADDR").getText());

　　　}

　　} catch (Exception e) {

　　　e.printStackTrace();

}

}

4）DOM4J

import java.io.\*;

import java.util.\*;

import org.dom4j.\*;

import org.dom4j.io.\*;

public class MyXMLReader {

　public static void main(String arge[]) {

　　long lasting = System.currentTimeMillis();

　　try {

　　　File f = new File("data\_10k.xml");

　　　SAXReader reader = new SAXReader();

　　　Document doc = reader.read(f);

　　　Element root = doc.getRootElement();

　　　Element foo;

　　　for (Iterator i = root.elementIterator("VALUE"); i.hasNext() {

　　　　foo = (Element) i.next();

　　　　System.out.print("车牌号码:" + foo.elementText("NO"));

　　　　System.out.println("车主地址:" + foo.elementText("ADDR"));

　　　}

　　} catch (Exception e) {

　　　e.printStackTrace();

}

)