文章编号:1005－0523(2004)01－0084－03

用SAX解析XML文档的实现方法

王 芳[1], 李正凡[2]

(华东交通大学 1.信息工程学院;2.软件学院,江西 南昌 330013)

摘要:讨论了解析XML文档的两种API.简要介绍了DOM和SAX在处理XML文档时的不同特点,重点分析了SAX的接口及其相关方法,并通过简单的示例展示了用SAX解析XML文档的方法.

关 键 词:XML;SAX;DOM;XML解析器
中图分类号:TP391.6 **文献标识码**:A

0 引 言

可扩展标记语言(eXtensible Markup Language, XML)是由万维网联盟(W3C)开发的,主要目的是为了克服超文本标记语言(Hypertext Markup Language, HTML)的缺点.作为HTML语言的一种补充,它具有可扩展,可移植,高度结构,简单,开放等一系列特性,从而受到业界的广泛关注,在WEB页上得到了广泛应用,成为Internet时代最有前景的标记语言.

近期内,XML主要应用在大型站点后台的维护工作,各组织间的信息交换,数据库的数据管理,电子商务和科学的应用程序,以及手持设备和智能电话等等,归纳起来也就是文档应用程序和数据应用程序两大类型.现在的主要软件厂商,如Microsoft, Netscape, IBM, Oracle等都在各自的产品中不同程度的支持XML,与此同时,基于各种语言的XML解析器纷纷出现了.不同于HTML,XML主要是描述数据本身而不是数据的表现格式,XML终于昭示了文档和数据是一回事,准确地讲,文档是数据的可互换形式[1].随着XML在WEB上的广泛使用,大量信息都要通过XML文档来存储,交换以及用XML表示各种应用接口,从而熟练掌握一种解析XML文档的方法是极为重要的.

文档对象模型(Document Object Model, DOM)和XML简单API(Simple API for XML,SAX)就是用来访问XML文档的API.通过它们的解析,应用程序读取XML文档而无需关心其语法.

1 XML简单API(SAX)及示例

XML程序由两部分组成:一是解析器,用于处理XML文件;二是应用程序,通过解析器对文件内容进行操作.而解析器和应用程序间的联系就是通过应用程序编程接口(Application Programming Interface, API),现今最流行的API就是如上所说的基于对象的API和基于事件的API,在实际的应用中,它们是互为补充的.

DOM是基于对象的接口:它通过在内存中明确地建立起一个对象树来和应用程序进行交互,其中包含了XML文档的所有元素,即这个对象树是对XML文档中元素树的精确映射.因此DOM的学习和使用都是比较简单的,对于像用于浏览器和编辑器的"XML密集型"[2]的应用程序,它是一个很好的选择.但是处理一般XML文档还只是众多任务中的一个,对于有自己的数据结构或是一个数据库的情

收稿日期:2003－09－28

作者简介:王 芳(1978－),女,湖北枝江人,硕士研究生,主要研究方向:数据库应用技术.

况，此时使用 DOM 模型并不合适，应用程序需要在内存中维持数据的两份拷贝（一份在 DOM 树中，而另一份在应用程序的数据结构中），对计算机内存要求很高，从而导致工作效率低下，严重时甚至导致系统崩溃．对于这种非“XML 密集型”的应用程序来说，SAX 极具优势，因为它不在内存中明确地建立文档树而是解析器从头到尾一次读完 XML 文档，每遇到一个解析事件就通知应用程序进行处理，这样对内存要求不高且速度很快．

在 XML 发展的初期，DOM 因是 W3C 认可的正式 API 而获得程序员的广泛使用．随着 XML 的发展，人们发现在解析的过程中，SAX 比 DOM 做的工作更少，效率却高了很多，层次也低了许多，于是人们逐渐从使用方便的 DOM 变为使用功能更为强大的 SAX，当然程序员就要做更多的工作了．SAX 最初是专为 JAVA 定义的接口，现在拥有 XML 处理程序的几乎所有语言都能支持 SAX，如 Python，Perl，C＋＋和 COM 等，通过 COM，所有的 Windows 编程语言都可以使用 SAX 解析器了．正是基于上述的原因，本文就重点对 SAX 展开讨论．

顾名思义，基于事件的解析器要向应用程序发送事件．在 XML 解析器中，事件并不与用户动作相关，而是与正在被解析的 XML 文档中的元素相关，解析过程中产生的主要事件有：元素的开始标记和结束标记；元素内容；实体；解析错误[2]．图 1 说明了解析器在读取文档时产生事件的过程：StartElement（“products”），StartElement（“product”），Characters（“xml study”），EndElement（“product”），StartElement（“price”）……

到目前为止，SAX 有两个版本，SAX1 和 SAX2，SAX2 在 SAX1 的基础上增加了对名称空间的处理，在本文中我们主要征对 SAX2 进行讨论．由于 SAX 是在面向对象的范例中设计的面向事件的 API，因此它与其他大多数时间驱动 API 不同，例如 expat 的 API，SAX 在单独的接口上组合了高度相关的处理程序[3]．所以学习 SAX 在很大程度上就是要掌握 SAX 的接口及相应的方法．SAX 接口主要有以下几类：

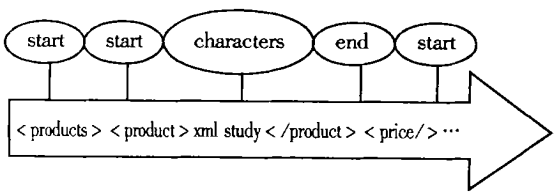

图 1　解析器生成事件

ContentHandler　是应用程序编写人员的核心接口，它实现了一些方法，定义了涉及文档自身的事件（如开始和结束标记），如上图 1 中产生的事件均由本接口提供；

DTDHandler　定义了涉及 DTD 的事件，但它还不能满足 DTD 的解析过程的全部需要，注意所有 DTDHandler 事件都发生于 startDocument 与第一个 startElement 事件之间；

DeclHandler　不是 SAX 的核心接口，而是一个对 DTDHandler 的扩展，它用来提供非数据 DTD 声明（如属性和属性列表声明）的信息；

ErrorHandler　定义了错误事件．解析器遇到错误时通知它，有时应用程序可以忽略错误并继续，有时则不行，应用程序必须发出错误消息；

XMLReader　是 SAX 中重要的接口，它是与 SAX2 兼容的分析程序必须实现的接口，以便于注册处理程序，从而获得和设置像名称空间这样的特性和属性，分析文档等，主要方法见表二；

EntityResolver　当解析器必须访问文档实体之外的文本时，该接口提供要调用的方法．

在上图 1 我们看到的解析器生成事件的过程中调用的方法 StartElement（），Characters（），EndElement（）均为 ContentHandler 接口的主要方法，除此以外它还有一些方法：StartDocument（），EndDocument（），IgnorableWhiteSpace（），ProcessingInstruction（），SetDocumentLocator（）等．StartDocument（）方法在文档开头调用一次（即遇到＜? xml version＝“1.0”? ＞时），让处理程序知道文档分析已经开始了；与此配套的 EndElement（）方法在元素结束时调用一次；IgnorableWhiteSpace（）方法通知处理程序有关连续可忽略的空白的情况，一次可能不会要求返回所有的空白，只需在该方法中加参数限制；ProcessingInstruction（）方法在每次发现处理指令时调用，注意某些处理指令可能发生在主文档外面，同时 SAX 分析程序还不能使用这个方法报告文本声明或 XML 声明；SetDocumentLocator（）方法由分析程序调用，用来设置以后发生的事件的定位器，从而提供生成各个事件的数据的本地化信息．这些方法都比较容易理解，也就不举例分析了．

由于本文篇幅有限，只能对 SAX 的重要接口进行分析，前面已对 SAX 的核心接口 ContentHandler 的主要方法进行了示例和分析，接下来我们看看下面这段代码对 SAX 的另一重要接口 XMLReader 是如何实现的（读者对照表 1 很容易理解下面这段代码）：

表 1 XMLReader 主要方法

Parse()	该方法解析一个 XML 文档，可以有两个不同的形式：一个接受文件名或 URL 作为参数，一个接受 InputSource 对象作为参数；
GetFeature()	该方法用来查询特性的状态(true 或 false)，各个特性是由一个完全合格的 URL 来标志的；
SetFeature()	该方法用来请求特性的状态更改，即控制解析器的工作方式，参数接受布尔值；
GetProperty()	该方法将返回使用 URL 名称标志的属性的当前值(如果有的话)；
SetProperty()	该方法控制解析器的工作方式，设置使用 URL 名称标识的属性值，参数接受对象.

```
XMLReader a = new MySAXDriver();
PricePrint myHandler = new PricePrint(); // PricePrint()为从 DefaultHandler 继承的类;
try {
  a.setFeature("http://xml.org/sax/features/validation", true);
  a.setProperty("http://xml.org/sax/features/DeclarationHandler", MyHandler);
  a.setContentHandler(new myHandler());
  a.setErrorHandler(new myHandler());
  }catch(SAXException e){
    System.err.println("XML exception setting handlers.");
  }
  try{
    a.parse("http://www.foo.com/mydoc.xml"); // parse()方法引发了对该 XML 文档的解析;
    }catch( IOException e){
    System.err.println("I/O exception reading XML document");
    }catch(SAXException e){
      System.err.println("XML exception reading document.");
    }
```

2 小 结

SAX 和 DOM 是时下最流行的两种解析 XML 文档的 API，考虑到 DOM 的易学性和 SAX 所具有的突出优点，本文重点讨论了 SAX 的特点及它所实现的接口，并对其重要接口的相关方法进行了较为详细的分析，同时通过简单的例子加以说明 SAX 解析 XML 文档的过程和方法.

参考文献：

[1] 王艳斌，赵伟明. XML 手册(第四版)[M]. 北京：电子工业出版社，2003.

[2] Benoît Marchal，XML 程序示例导学(第二版)[M]. 北京：清华大学出版社，2002.

[3] Fabio Arciniegas，XML developer's guide 2003，The McGraw—Hill Companies，Inc.

The Realization Method of Parsing XML Document by SAX

WANG Fang[1]，LI Zheng-fan[2]

(East China Jiaotong University 1. School of Information Engineering; 2. School of Software，Nanchang Jiangxi 330013，China)

Abstract: This paper discusses the two kinds of API of parsing XML documents，introduces the different characters about DOM and SAX，analyses the interfaces and methods of SAX and brings forth the methods of how to parse an XML document by SAX through simply examples.

Key words: XML; SAX; DOM; XML Parser