JAVA 技术丛书

Java语言与XML处理教程

SAX, DOM, JDOM, JAXP与TrAX指南

Processing XML with Java

A Guide to SAX, DOM, JDOM, JAXP and TrAX

[美] Elliotte Rusty Harold 著
刘文红 赵伟明 等译

PEARSON Addison Wesley
PHEI
電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

Java 技术丛书

Java 语言与 XML 处理教程

SAX，DOM，JDOM，JAXP 与 TrAX 指南

Processing XML with Java

A Guide to SAX, DOM, JDOM, JAXP and TrAX

[美] Elliotte Rusty Harold 著

刘文红 赵伟明 等译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

本书是为要在系统中集成XML的Java编程人员编写的，是介绍如何使用Java编程语言处理XML文档的实用而且综合的指南与教程。书中简要概述了XML基础，包括XML语法、DTD、模式、有效性，样式单和XML协议XML-RPC、SOAP与RSS。本书的核心内容是深入介绍了Java编程人员用Java生成与操纵XML文档时所用的关键XML API，包括SAX、DOM（文档对象模型）和JDOM。此外，还介绍了这些核心API的许多重要补充，包括XPath，XSLT，TrAX与JAXP。

本书详尽介绍了实用和面向任务的方法，是所有需要使用XML的Java编程人员的宝贵参考资料。

图书在版编目（CIP）数据

Java语言与XML处理教程：SAX, DOM, JDOM, JAXP与TrAX指南/（美）哈罗德（Harold, E. R.）著；刘文红等译. -北京：电子工业出版社，2003.11
（Java技术丛书）
书名原文：Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP and TrAX
ISBN 7-5053-9277-8

Ⅰ.J... Ⅱ.①哈... ②刘... Ⅲ.①Java语言-程序设计 ②可扩充语言，XML-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第098435号

责任编辑：窦 昊　　特约编辑：赵宏英
印 刷 者：北京兴华印刷厂
出版发行：电子工业出版社 http://www.phei.com.cn
北京市海淀区万寿路173信箱 邮编：100036
经 销：各地新华书店
开 本：787 × 1092 1/16　印张：42.25 字数：1082千字
版 次：2003年11月第1版　2003年11月第1次印刷
定 价：68.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换；若书店售缺，请与本社发行部联系。
联系电话：（010）68279077。质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

前　言

一天晚上，五位新加入Elephants公司的开发人员坐在一起熟悉公司的订单处理系统。他们钻进主服务器中一大堆XML文档的目录中。“这是什么?”小组长激动地问。他们过去谁也没有听说过XML，因此准备了解手头的文件，看看这个奇怪而奇妙的新技术到底是怎么回事。

第一位开发人员擅长于Oracle数据库优化，因此打印了一堆FileMaker数据库产生的FMPXMLRESULT文档，开始进行分析，这个数据库中存放了所有订单。“这就是XML，我明白了，也没什么新东西。不难看出，XML文档不过就是表格”。

“什么，就是表格？”第二位开发人员说，他非常熟悉面向对象理论，手头有一堆XMI文档对系统的UML框架进行了编码。“即使Visual Basic编程人员也能看出，XML文档不是表格，表格关系中不允许复制，除非这是些奇怪的变种。这些XML文档明摆着就是类和对象。事实上，这是一目了然的。XML文档是对象，DTD是类”。

“对象？真是奇怪的对象”，第三位开发人员说，他是一位具有一定知名度的Web设计人员，把订单处理系统的XHTML用户文档装入Mozilla中。“我没有看到任何类型。如果你认为这是对象，我拒绝安装你的软件。但从这么多样式单不难看出，XML就是更新的HTML”。

“HTML？别开玩笑”，第四位说，他是来自MIT的计算机科学教授，曾经致力于用XSLT样式单和Schematron模式检验所有其他文档。“看看层次结构的整洁嵌套，每个标志与相应标志恰好匹配，HTML什么时候有这么漂亮。这是个S表达式，没什么新东西，1882年就已经出现这个思想了”。

“S表达式？”技术作家问，他手头有一大堆用DocBook编写的项目文档。“对你们这些博学之士，也许是这样吧。但是我觉得这就像是FrameMaker MIF文件。不过熟悉GUI好像有点费事”。

他们就这样相互争吵，直到深夜，谁也不服谁，每个人都引经据典，却又都不屑看看别人的证据。事实上，他们可能现在还在争论，你甚至还能听到他们的争吵声。当然，他们的错误在于执意把XML归结到他们已经熟悉的技术，而不是把XML当做全新的技术。XML可以存储数据，但不是数据库。XML可以将对象序列化，但XML文档不是对象。XML可以编写Web页面，但XML不是HTML。XML可以编写功能性（和其他）编程语言，但XML本身不是编程语言。XML可以写书，但XML不是桌面出版软件。

XML是个新东西，在计算世界中没有先例。XML继承了过去的思想，一些固执的人总想通过数据库、对象、功能性或S表达式之类的有色眼镜来看XML，但XML并不是这些东西，而是个新东西，是一个在计算世界中没有先例的新概念。要理解XML，首先要把它看成一个新概念，而不是跨进昨天的旧框框中。

世界上已经有许多工具、API和应用程序，使XML看上去像是开发人员熟悉的某个旧东西——像某种数据库、某种对象或远程过程调用。这些API在限制性的可预测环境中也许有用，

但不适合处理更一般格式的XML。这些API在有限领域中能够顺利工作，但超出人为定义的边界时就行不通了。XML是可扩展的，但许多为XML设计的工具并没有这么强的可扩展性。

本书要介绍如何处理完全一般性的XML，而不是局限于某一个方面。XML就是XML，而不是任何别的东西。我们介绍如何设计程序，处理真正意义上的XML：有效与无效、混合与非混合、类型与非类型，以及这些特性的不同组合。为了达到这个目标，本书主要考虑不会隐藏XML本质的API。特别地，有三大类Java API正确地建模XML，而不是建模XML文档的特定类或XML的某个狭窄子集。这些Java API分别是：

- SAX（XML的简单API，Simple API for XML）
- DOM（文档对象模型，Document Object Model）
- JDOM，Java本地API

这些API是本书的核心。此外，书中还将介绍基本API的几个前提与补充项目，包括：

- XML语法
- DTD、模式与有效性
- XPath
- XSLT与TrAX API
- JAXP是SAX，DOM，TrAX和几个工厂类的结合

由于书中要用几个XML应用程序例子演示这些API，因此还要详细介绍XML-RPC, SOAP与RSS。但是，本书介绍的技术不局限于这三类应用程序。

适用读者

本书是为要在系统中集成XML的高级Java开发人员编写的。Java是处理XML文档的理想语言，特别是其强大的Unicode支持使其成为许多早期实现版本的首选语言。因此，用Java编写的XML工具比用其他任何语言编写的XML工具都多，用Java编写的开放源代码XML工具也比用其他任何语言编写的都多。大多数开发者都用Java处理XML。

本书主要介绍下列内容：

- 从Java应用程序中保存XML文档
- 读取其他程序产生的XML文档
- 搜索、查询与更新XML文档
- 将遗留平面数据转换成层次式XML格式
- 与发送和接收XML数据的网络服务器通信
- 用DTD、模式和业务规则验证文档
- 组合功能性XSLT转换与传统的命令式Java代码

本书适用于用XML进行任何工作的Java开发人员，全面介绍了基础课题和高级课题，是Java语言与XML处理教程的综合性课程，使不熟悉XML的开发人员可以通过学习本书设计出

复杂的XML应用程序和分析复杂的XML文档。本书的例子中包括各种可能的应用，包括文件格式、数据交换、文档转换、数据库集成，等等。

预备知识

本书不是Java与XML的入门书籍，要求读者对Java与XML有一定的了解。在Java方面，书中随意使用这个语言的高级特性及其类库，不做太多解释。读者至少要熟悉下列内容：

- 面向对象编程，包括继承与多态。
- 包与CLASSPATH，要习惯于没有main( )方法的类和不在默认包中的类。
- I/O包括流、阅读程序和写入程序，要了解System.out是Java程序中很常用的工具。
- Java Collections API包括散列表、映射、集合、迭代器和列表。

此外，本书某些地方还用到SQL与JDBC。但这些部分在书中相对独立，如果读者不熟悉SQL，则通常不需要这些材料。

所需工具

XML是独立于体系结构、平台、操作系统、GUI和语言的（比Java更加独立），在Mac OS, Windows, Linux, OS/2和各种UNIX系统中都能顺利工作，可以用Python, C++, Haskell, ECMAScript, C#, Perl, Visual Basic, Ruby和Java等语言访问。在PowerPC, X86和其他体系结构之间进行切换时，不需要考虑字节顺序问题。几乎本书的任何内容在任何能运行Java的平台中都同样适用。

本书的大部分材料相对独立于特定的Java版本。Java 1.4把SAX, DOM和另外几个重要类捆绑到核心JDK中，但它们在旧版JVM中也很容易安装，因为Apache XML Project和其他厂家提供了开放源代码库。通常，书中在测试例子时使用Java 1.3和1.4，因此所用的有些类与方法可能在旧版中没有。但是在大多数情况下它们很容易向后移植。除TrAX以外的所有基本XML API都适用于Java 1.1以上版本，TrAX适用于Java 1.2以上版本。

本书用法

本书编写成高级教程，也可以作为综合性的参考手册。第1章介绍使用XML所需的最基本材料，但不作为综合性引言，而是作为复习材料，要求读者先阅读一些更基础的书籍。第2章介绍RSS，XML-RPC，SOAP和本书例子中使用的XML应用程序。后面两章介绍从程序中产生XML（这个课题常常被介绍得过于复杂）。第3章介绍从代码中直接生成XML，第4章介绍将其他格式的遗留数据转换成XML。本书其余内容介绍几个主要的API，用于处理XML：

- 基于事件的SAX API
- 基于树的DOM API
- 基于树的JDOM API

- 搜索 XML 文档的 XPath API
- 用于 XSLT 处理的 TrAX API

本书最后的附录提供主要 API 的速查手册。

如果读者的 XML 经验不是太丰富，建议至少先依次读完本书前两章。然后如果喜欢特定的 API，可以直接转入相应部分：

- 第 6 章到第 8 章介绍 SAX。
- 第 9 章到第 13 章介绍 DOM。
- 第 14 章到第 15 章介绍 JDOM。

熟悉一个或几个 API 之后，可以学习第 16 章与第 17 章介绍的 XPath 与 XSLT。但是，这些章节和 API 的内容要求至少对这三大 API 之一有一定了解。

联机版本

可以从笔者的Cafe con Leche站点取得本书的联机版本（原英文版），网址为http://www.cafeconleche.org/books/xmljava/，本书的每一个字都在上面，毫无保留。我希望你喜欢印刷版本，买上一本，比你自己用激光打印机打印1120页的原书成本要低得多，但这完全凭你自愿。我的目标是尽量推广本书内容，使其为更多人服务。

联机版本受到版权法的保护，阅读时不需要注册，也不必下载特殊的电子键而在换一台电脑时失效。我希望更多人阅读本书，使用其中的知识。作为礼节，希望读者不要在自己的服务器上重新发表这个联机版本，否则很难让你的副本保持更新。如果你想在手提电脑中保存几页，以便在飞机上阅读，我不会介意，但不要把自己的副本发给别人，而要让你的同事和朋友到我的 Web 站点或者买一本书。

联系作者

我喜欢听到读者的意见、建议和与本书有关的问题。由于本书完全联机发表，联机版本可以比传统图书更快改版，因此纠正与勘误特别有用，我可以随时修正。发来纠正与勘误之前，请先看看联机版本是否已经解决了所提的问题。

请将所有意见、建议和问题发到elharo@metalab.unc.edu。由于收到了很多E-mail，因此不能保证有信必复，但我会尽力而为。最好能在主题中明确表示你是本书读者，否则你的消息很容易被当做垃圾邮件，我可能不加浏览就扔进垃圾箱中。另外，请确保你的邮件使用正确的答复地址，至少在发信后的几个月内保证这个地址有效。最令人失望的是，花好几个小时写了一封信之后，却因为地址不对而被退回。欢迎来信，盼望你的意见、建议和问题。

目　录

第一部分　XML

第二部分 SAX

第三部分 DOM

第四部分 JDOM

第五部分 XPath/XSLT

第六部分 附 录

第一部分
XML

第1章 XML与数据

XML原先只是“Web上的SGML（标准通用标记语言）”，最初和SGML与HTML一样在同一种叙述性文档中使用：文章、图书、故事、诗歌、技术手册、Web页面，等等。但是，让XML发明者大为吃惊的是，XML的巨大成功不是在写作与发表领域，而是在更为枯燥乏味的数据格式世界中。通常，这不像文章与故事之类的叙述性数据，而是面向记录的数据，如数据库数据，用于对象序列化、财务记录、矢量图形、远程过程调用等任务。本章介绍这些数据传统格式中的一些缺陷，看看XML的特性是如何恰好适合这些任务的。

1.1 XML的优点

如果你是位开发人员（至少我希望是，否则书中的内容可能让你没兴趣），毫无疑问，你在工作中已经编写了许多读取与写入文件的程序。每次编写新程序时，都要发明或学习新文件格式。几年来，我处理过的文件格式包括RTF, Word .doc文件，制表符分隔文件，FITS, PDF, PostScript，等等。毫无疑问，你一定还遇到过许多其他格式。

每次遇到新文件格式时，总是件麻烦事。如果这个文件格式有文档，通常也是文档不全，甚至可能错误连篇，通常没有指定字节顺序、行末规则等重要细节。号称能够读取与写入同一种文件格式的不同工具实际上会产生稍有不同的变体，实际应用时常常互不兼容。当你以为自己终于消灭了代码中最后一处缺陷时，却可能发现无法阅读别人的软件写成的文件。你发现自己对文件格式做了太多假设，又要回到草稿，重新开始。

因此，设计新文件格式时，开发人员通常采用最简单的格式，通常是制表符分隔文本或逗号分隔值。然而，即使如此简单的无修饰格式也常常遇到意想不到的问题。例如，把行中的两个制表符解释为空字符串、null还是等于一个制表符？实际应用中，这几种变体都在使用。Java语言的StringTokenizer类采用最后一种解释：连续两个制表符等同于一个制表符，但这只是实际数据文件中的最不常用方法，使许多Java编程人员大为吃惊，在Java程序中带来不少缺陷[①]。

1.1.1 一个思想试验

了解上述情况后，下面来做一个思想试验。假设任务是编写一个服务器方程序，通过Internet接受一个电子商务站点的订单。Web服务器要将每个完成的订单发送到内部系统中，一次发送一个订单。你负责编写服务器上的代码，将订单发送到内部系统，并负责编写内部系统上的代码，接收和处理订单。这两个系统间惟一的连接是TCP/IP网络，即没有JDBC之类的高级API帮你在两个系统之间移动数据。你要发明一种数据格式，在一端产生，在另一端分析，

① 要理解这一点，注意java.util.StringTokenizer的设计目标是分析Java源代码，而不是读取制表符分隔的数据文件。但许多Java编程人员用它读取制表符分隔数据文件。

这种数据格式应当足够灵活以包含典型订单中的所有信息，包括客户名、所订产品、价格、制造商的库存单元（SKU）数、发货地址、税收和发货与处理费用。一种方法是将每个信息放在一行，如例子 1.1 所示。

例子 1.1　表示 12 Birdsong Clocks, SKU 244 订单的普通文本文档

```
c32
Chez Fred
Birdsong Clock
244
12
USD
21.95
135 Airline Highway
Narragansett
RI
02882
USD
263.40
7.0
USD
18.44
USPS
USD
8.95
USD
290.79
```

另一种方法是使用更复杂更冗长的 XML 格式，如例子 1.2 所示。

例子 1.2　表示 12 Birdsong Clocks, SKU 244 订单的 XML 文档

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Order>
  <Customer id="c32">Chez Fred</Customer>
  <Product>
    <Name>Birdsong Clock</Name>
    <SKU>244</SKU>
    <Quantity>12</Quantity>
    <Price currency="USD">21.95</Price >
  </Product>
  <ShipTo>
    <Street>135 Airline Highway</Street >
    <City>Narragansett</City> <State>RI</State> <Zip>02882</Zip>
  </ShipTo>
  <Subtotal currency='USD'>263.40</Subtotal>
  <Tax rate="7.0"
       currency='USD'>18.44</Tax>
  <Shipping  method="USPS" currency='USD'>8.95</Shipping>
  <Total currency='USD' >290.79</Total>
</Order>
```

编写发送与接收订单的代码时，是用例子 1.1 所示的简单分行符分隔文件还是用例子 1.2 所示的复杂标记 XML 格式？这两个文档包含相同信息。大多数没经验的开发人员喜欢第一种简

单格式，因为每个信息另起一行，不必增加多余的标识符。我的目的是要说明，与大多数开发人员的最初直觉相反，第二种格式更健壮、更可扩展和更容易使用。

1.1.2 健壮性

先看看健壮性。假设程序接收例1.3所示的订单。

例子1.3 表示12 Birdsong Clocks, SKU 244订单的文档

```
c32
Chez Fred
Birdsong Clock
12
244
USD
21.95
135 Airline Highway
Narragansett
RI
02882
USD
263.40
7.0
USD
18.44
USPS
USD
290.79
USD
8.95
```

它看上去和例子1.1相同，但如果认真比较，则可以发现12与244的位置互换了，原先12个报时钟的订单变成244个坐垫的订单。这个问题也许能在订单发货之前被发现，也许不能在订单发货之前被发现。更糟的是，发货费与总价也改变了，整个订单的成本变成8.95美元。这个问题也许能及时发现，也许不能及时发现。这类问题是实实在在的，许多电子商务站点因为价格错误而失去收入和信誉。

在XML版本中，根本不会有这个问题，因为每个数据项目都标明含义，可以随意打乱数量与SKU、发货成本与总价的顺序，不会产生任何混乱，如例子1.4所示。传统系统中可能铸成大错的问题在XML中不成问题。

例子1.4 另一个表示12 Birdsong Clocks, SKU 244订单的文档

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Order>
  <Customer id="c32">Chez Fred</Customer>
  <Product>
    <Name>Birdsong Clock</Name>
    <Quantity>12</Quantity>
    <SKU>244</SKU>
    <Price currency="USD">21.95</Price >
  </Product>
  <ShipTo>
    <Street>135 Airline Highway</Street >
    <City>Narragansett</City> <State>RI</State> <Zip>02882</Zip>
```

```
  </ShipTo>
  <Subtotal currency='USD'>263.40</Subtotal>
  <Tax rate="7.0"
       currency='USD'>18.44</Tax>
  <Total currency='USD' >290.79</Total>
  <Shipping  method="USPS" currency='USD'>8.95</Shipping>
</Order>
```

也许有人会说自己的系统中不会出现这类错误，因为他会检查每个值的意义。你会检查公司数据库中的SKU，保证其符合产品名和价格，然后再完成订单。也会检查每个方法调用的返回值，看看其是否为null，并捕获每个异常。你会编写大量测试，验证每个方法正常工作。会用源代码控制系统，因此总是可以撤销改变，代码要在进行所有回归测试并成功之后才能准入。每一行代码都文档齐全。事实上，所写的文档比实际代码还多。

即使你如此慎重，你能保证其他人发送与接收数据时也能同样慎重吗？倒不如采取更健壮的格式，使系统在遇到不可避免的错误时，不会造成更大的损害。

当然，XML对顽固的开发人员也大有帮助。定义限制时，如“每个订单要有一个发货地址”，“币值单位应为三字符代码USD, CAN或GBP”或“总价应为单价与数量的积加上税收和发货成本”，一种最简单的方法是使用声明式语言，指定所要的限制，而不是通过实际代码检查这些限制。对于XML格式数据，可以用声明式模式语言定义和测试这些限制。事实上，可以选择几种模式语言。最简单和最广泛支持的是经典的DTD（文档类型定义），可以检验所要的元素是否全部按所需的顺序出现，并且具有任何必要的属性。W3C XML模式语言更进了一步，可以限制特定元素与属性的内容，保证总价是大于1.00的十进制数。Schematron是最强大的模式语言，可以指定多元素限制，如“实际价格应小于或等于建议零售价”。

本章稍后和本书其余部分将详细介绍所有这些语言。目前只要知道，可以用简单格式在文档中列出所有限制，然后检查这些限制，而不需要大量多余的代码。可以先用验证程序检查文档之后再对其进行操作。验证是单独的、模块化和更易维护的过程。甚至可以不重新编译代码而改变限制或增加新限制。

1.1.3　扩展性

健壮性不是XML方法的惟一好处。XML还具有更好的扩展性。例如，假设突然需要在一些产品中增加折扣百分比。XML代码很容易改变，只要增加另一个元素：

```
<Product>
  <Name>Birdsong Clock</Name>
  <Quantity>12</Quantity>
  <SKU>244</SKU>
  <Price currency="USD">21.95</Price >
  <Discount>.10</Discount>
</Product>
```

而改变普通文本文件（或相应的二进制文件）则麻烦得多。尽管可以增加一行数据，但这样会使后面的所有内容顺序出错。可以将新信息放在文档末尾，但这样就不能让它靠近逻辑上归属的项目。假设并不是所有订单都有折扣，则没有折扣的产品是否使用空行？程序怎么知道空字符串要转换成0折扣，而不是转换成NaN（Not a Number）或抛出异常？这个问题不是无法解决，但会把简单问题复杂化。