JAVA 技术丛书

Sun microsystems

Java Web Services教程

The Java Web Services Tutorial

[美] Eric Armstrong 等著

詹文军 廖 铮 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 Web Services 教程

The Java Web Services Tutorial

[美] Eric Armstrong 等著

詹文军 廖 铮 等译

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

内容简介

本书是一本内容全面的Java Web Services的指导书，作者均为Sun公司的高级专业技术人员和技术文档编写人员。书中通过大量实例指导读者使用Java技术创建Web服务应用程序。本书介绍了如何使用Sun公司的WSDP（Java Web Services Developer Pack）这一内容丰富、易于使用的程序包，讲解了各种用于创建和部署Web服务应用的技术和工具。此外，本书还详细介绍了API，并提供了一些例子供读者练习以加深对关键概念的理解。本书适用于利用Java技术的专业人员和广大用户。

版权贸易合同登记号：图字：01-2002-5200

图书在版编目（CIP）数据

Java Web Services 教程/（美）阿姆斯特朗（Armstrong, E.）著；詹文军等译. -北京：电子工业出版社，2003.8
（Java 技术丛书）
书名原文：The Java Web Services Tutorial
ISBN 7-5053-8917-3

Ⅰ.J... Ⅱ.①阿... ②詹... Ⅲ.JAVA语言-程序设计-教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第061189号

责任编辑：周宏敏
印 刷 者：北京四季青印刷厂
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：22.25 字数：570千字 附光盘1张
版 次：2003年8月第1版 2003年8月第1次印刷
定 价：39.00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

译 者 序

《Java Web Services 教程》是 Javs Series 系列图书中的一本，该系列图书由 Sun 公司的 Java 技术创始人员参与了编写、审订以及提供技术支持，用该系列图书总编辑 Lisa 的话说，该系列书是“来自源头的权威图书”。既然是好书，作为译者的我们也是不敢怠慢，在翻译过程中全力以赴，生怕由于自己的工作不到位而坏了读者的胃口。好在本书的光盘中提供了该书的原版内容，另外，在 Java 公司的站点上也可以下载本教程，因此，如果读者愿意，可以直接阅读原版。但是我们建议读者先阅读中文版，一来我们希望通过自己的付出能够为读者减少阅读原版所费的时间和精力，二来也是请读者对我们的工作进行监督。读者若有疑问，请参照原版，如果读者觉得是我们译得不对，欢迎按以下地址将您的意见发给我们：zhanwenjun@263.net。如果读者觉得是原书的问题，请阅读“前言”并给 Lisa 发邮件。

由于本领域的一些术语叫法不一，我们在翻译本书时参考了一些较普遍的说法，此外，对于书中个别一些没有很好说法的术语，我们保留了原来的说法（例如 messaging provider，profile 等）。

参加本书翻译的有詹文军、廖铮、王强、孙淳，全书由詹文军负责统稿，卢陈思参与了本书的录入工作。

序

Java技术早在1996年的秋季就开始吸引了人们对它的关注和热情，但是在Sun Microsystem公司，我们的小组还是一个规模很小的小组，那时我们只有25人，我们只好轮流回复那些开始如潮水般涌入我们信箱的邮件。

即使在1996年，就有许多开发人员问我们："有关Java内容的最佳图书有哪些？"许多人还给我们提建议说："我们很喜爱由Campione和Walrath所编写的在线Java教程，这些教程什么时候能作为图书面市呢？"

这使得我们意识到在我们倡导在线出版技术文档这一理念的同时，我们的小组还需要成为有关Java这一新技术的图书海洋中的"灯塔"。因此，我和James Gosling和Bill Joy一起启动了Java Series系列图书的出版工作，该系列图书直接由我们的工程小组成员所编写，并由Addison-Wesley公司代表我们出版。开发人员具有大量的图书可供选择，而且如果他们需要，可以直接从Java技术的"源头"取经。

随着Web Services技术的出现，当年人们对Java的狂热又再次重现。我们收到的电子邮件表明，开发人员目前迫切需要有关这一技术的教程。尤其对于先期访问的技术，开发人员更是希望听到"来自源头的权威声音"。

因为七年来我们曾编写了有关Java技术的权威性教程，因此读者完全可以信赖这一事实：我们的作者、编辑和技术评审知道如何将原先应用于Java教程的高标准要求应用到关于Web Service的图书上。

Web Services教程的这一Early Access（先期访问）版本是为了迎接2002年度的JavaOne开发者大会而应时发布的：我们得到的信息是：现在，你需要使用Web Services技术才能获得成功。在本书附带的CD中包括了所有的下载内容、教程、演示和示例代码，读者可以通过它们来帮助你对Web Services技术迅速上手。

读者对本书的反馈可以发送到webservicestutorial@sun.com。正如我们在1996年所做的那样，我们会读取并回复每封关于Java系列图书的邮件。因此你可以将你的想法写给我们，我们保证会给你提供答复。

Lisa（Java Series系列图书总编）
Sun Microsystems, Inc.
Santa Clara, CA

前　言

本书是一本关于使用Java Web Services开发包（Java WSDP）来开发Web服务和Web应用程序的初级指南。Java WSDP是一个一体化的下载开发包，其中包含了使用Java 2平台来简化构建Web服务的关键技术。其中包括以下内容：

- Java Servlets
- JavaServer Pages（JSP）
- JSP Standard Tag Library（JSTL）
- Java XML Pack，它包括：
  - Java API for XML Messaging（JAXM）
  - Java API for XML Processing（JAXP）
  - Java API for XML Registries（JAXR）
  - Java API for XML-based RPC（JAX-RPC）

为提供一个开发和部署环境，Java WSDP还包括以下内容：

- Tomcat servlet和JSP container
- Ant build工具
- Java WSDP Registry Server

我们将在前言中介绍一些必备内容，以便更好地帮助读者阅读本书。

读者对象

本教程面向那些对使用Java WSDP来开发和部署Web服务以及Web应用程序的程序员。

关于示例

本教程包括了许多完整而实用的示例。

理解示例的先决条件

为理解本教程中的示例，需要对Java编程语言、SQL和关系数据库概念有很好的了解。尤其是与Java教程中的以下主题相关的内容格外重要：

主题	教程
JDBC	http://java.sun.com/docs/books/tutorial/jdbc
线程	http://java.sun.com/docs/books/tutorial/essential/threads
JavaBeans	http://java.sun.com/docs/books/tutorial/javabeans
安全性	http://java.sun.com/docs/books/tutorial/security1.2

运行示例

这部分告诉你为获得、连编、部署以及运行示例所需要了解的内容。

需要的软件

如果是在线阅读本教程，则需要从以下地址下载本书内容：

```
http://java.sun.com/webservices/downloads/webservicestutorial.html
```

一旦安装了教程程序包，则示例程序的源代码位于<JWSDP_HOME>/docs/tutorial/examples目录中，该目录下为程序包中所包括的每种技术都建有一个子目录。

本教程是关于 Java WSDP EA1 的技术文档，为连编、部署并运行示例，需要有 Java WSDP 和 Java 2 Platform，Standard Edition（J2SE™）SDK 1.3.1 或 1.4。读者可以从以下地址下载 Java WSDP：

```
http://java.sun.com/webservices/downloads/webservicespack.html
```

从以下地址下载 J2SE 1.3.1 SDK：

```
http://java.sun.com/j2se/1.3/
```

或从以下地址下载 J2SE 1.4 SDK：

```
http://java.sun.com/j2se/1.4/
```

读者可以按照表 1 中的说明设置有关环境变量。

表 1　需要的环境变量

环境变量	设置值
JAVA_HOME	J2SE SDK 安装程序的所在位置
JWSDP_HOME	Java WSDP 安装程序的所在位置。该变量由示例 build（连编）文件使用
PATH	将 Java WSDP 和 J2SE SDK 的安装程序所在目录的 bin 目录加入其中，Java WSDP bin 目录包含了针对 Tomcat、ant、registry server（注册表服务器）以及其他工具的启动脚本

连编示例

大多数示例都在分发时附带了 ant 1.4.1 版本的一个配置文件，ant 是在 Java WSDP 中包含的一个可移植 build 工具。在每章中都提供了对示例程序进行连编（build）的指导。

部署示例

大多数Java WSDP 示例运行于Tomcat。在运行某个示例之前，必须首先将它部署到Tomcat。为部署某个应用程序，可以执行 ant deploy。部署时通常会把一些文件复制到<JWSDP_HOME>/webapps 目录。以下两点需要注意：

- 对于此版本的 Java WSDP，必须在开发应用程序的同一计算机上运行 Tomcat。
- 当应用程序第一次被部署时，必须启动或重新启动 Tomcat，之后，当对应用程序进行修改后，可以对它进行连编、部署，然后重新加载。

运行 Tomcat

可以通过在某个终端窗口中执行 startup 脚本来运行 Tomcat。

重新加载示例

可以使用以下命令来重新加载 一个应用程序：

```
http://localhost:8080/manager/reload?path=/target
```

该命令调用了 manager 这个 Web 应用程序，在使用该应用程序之前，必须将你的用户名和密码组合与角色名 manager 进行关联，并加入到<JWSDP_HOME>/conf/tomcat-users.xml，它可以使用任何文本编辑器来编辑。该文件为每个单独的用户包含了一个<user> 元素，其形式如下所示：

```
<user name="adeveloper" password="secret" roles="manager" />
```

随 Java WSDP 一起分发的 Tomcat 参考文档包含了有关 manager 应用程序的内容。

相关内容

有关在本教程中所讨论的技术的深入内容，可以参见 Java WSDP 中所包含的参考文档（<JWSDP-HOME>/docs/index.html）和表 2 中列出的 Web 站点。对本教程一些章节中所列出的个别技术主页的引用方式表示如下：

- JAXM_HOME 指的是 JWSDP_HOME/docs/jaxm/index.html
- JAXP_HOME 指的是 JWSDP_HOME/docs/jaxp/index.html
- JAXR_HOME 指的是 JWSDP_HOME/docs/jaxr/index.html
- JAXRPC_HOME 指的是 JWSDP_HOME/docs/jaxrpc/index.html

表 2　相关内容

技术	Web 站点
Java Servlets	http://java.sun.com/products/servlet/index.html
JavaServer Pages（JSP）	http://java.sun.com/products/jsp/index.html
JSP Standard Tag Library（标准标签库）	http://java.sun.com/products/jsp/taglibraries.html#jstl
JAXM	http://java.sun.com/xml/jaxm/index.html
JAXP	http://java.sun.com/xml/jaxp/index.html
JAXR	http://java.sun.com/xml/jaxr/index.html
JAX-RPC	http://java.sun.com/xml/jaxrpc/index.html
Tomcat	http://jakarta.apache.org/tomcat/index.html
ant	http://jakarta.apache.org/ant/index.html

如何打印本教程

要打印本教程，可以按照以下步骤操作：

- 确保你的计算机上安装了 Adohe Acrobat Reader 程序。
- 打开本书的 PDF 版本。
- 单击 Adobe Acrobat Reader 程序中的打印机图标。

目　录

第 1 章　Web 服务介绍 1
1.1　XML 和 Java 平台的角色 1
1.2　什么是 XML 2
1.3　Java API for XML 概述 4
1.4　JAXP 5
1.5　JAX-RPC 10
1.6　JAXM 11
1.7　JAXR 16
1.8　示例 18

第 2 章　了解 XML 20
2.1　XML 介绍 20
2.2　XML 和相关规范 26
2.3　设计 XML 数据结构 33

第 3 章　从 Tomcat 开始 38
3.1　设置 38
3.2　创建 Getting Started 应用程序 39
3.3　使用 ant 连编和部署 Getting Started 应用程序 41
3.4　运行 Getting Started 应用程序 43
3.5　修改应用程序 44
3.6　常见问题及其解决方案 45

第 4 章　JAXP 47
4.1　JAXP API 47
4.2　程序包概述 47
4.3　SAX API 48
4.4　DOM API 50
4.5　XSLT API 51
4.6　编译并运行程序 52
4.7　进一步的阅读 52

第 5 章　SAX 54
5.1　编写一个简单的 XML 文件 55

5.2 定义根元素 55
5.3 使用SAX解析器回显一个XML文件 58
5.4 添加额外的事件处理程序 68
5.5 使用非验证型解析器处理错误 70
5.6 替代和插入文本 76
5.7 创建一个DTD 79
5.8 DTD对非验证型解析器的影响 82
5.9 定义DTD中的属性和实体 84
5.10 引用二进制实体 89
5.11 使用验证型解析器 90
5.12 定义参数实体和条件段 93
5.13 对参数化DTD进行解析 95
5.14 处理词法事件 97
5.15 使用DTDHandler和EntityResolver 102

第6章 文档对象模型 104
6.1 把XML数据读取到DOM中 104
6.2 显示DOM层次结构 109
6.3 检查DOM的结构 120
6.4 从DOM创建一个用户友好的JTree 124
6.5 创建并操作DOM 135
6.6 使用名字空间 139

第7章 XSLT 142
7.1 介绍XSLT和XPath 142
7.2 将DOM作为XML文件写出 148
7.3 从数据结构生成XML 153
7.4 使用XSLT转换XML数据 162
7.5 使用一个过滤器链串接XSLT转换 179

第8章 JAXM 185
8.1 JAXM概述 185
8.2 运行示例 190
8.3 教程 192
8.4 代码示例 204

第9章 JAX-RPC 213
9.1 什么是JAX-RPC 213
9.2 一个简单示例: HelloWorld 214
9.3 动态调用接口 221

第 10 章 JAXR 225
10.1 JAXR 概述 225
10.2 实现一个 JAXR 客户 227
10.3 使用注册表浏览器 235

第 11 章 Java WSDP 注册表服务器 239
11.1 设置注册表服务器 239
11.2 通过注册表浏览器使用注册表服务器 240
11.3 通过注册表服务器使用命令行客户脚本 240
11.4 使用 JAXR API 访问注册表服务器 241
11.5 使用 Indri 工具访问注册表服务器数据库 242

第 12 章 Web 应用程序 244
12.1 Web 应用程序的生命周期 244
12.2 Web 应用程序档案 246
12.3 Web 应用程序部署描述符 246
12.4 部署 Web 应用程序 249
12.5 运行 Web 应用程序 250
12.6 更新 Web 应用程序 250
12.7 对 Web 应用程序进行国际化和本地化 251
12.8 从 Web 应用程序访问数据库 252

第 13 章 Java Servlet 技术 255
13.1 什么是 servlet 255
13.2 示例 servlet 256
13.3 servlet 的生命周期 257
13.4 共享信息 259
13.5 初始化 servlet 261
13.6 编写服务方法 262
13.7 过滤请求和响应 266
13.8 调用其他 Web 资源 270
13.9 访问 Web 上下文环境 273
13.10 维护客户状态 273
13.11 结束一个 servlet 275

第 14 章 JSP 技术 278
14.1 什么是 JSP 页面 278
14.2 JSP 页面示例 280
14.3 JSP 页面的生命周期 281
14.4 初始化和结束 JSP 页面 283

14.5 创建静态内容 284
14.6 创建动态内容 284
14.7 在 JSP 页面中包括内容 288
14.8 将控制权转移到其他 Web 组件 289
14.9 包括小应用程序 289
14.10 扩展 JSP 语言 291

第 15 章 JSP 页面中的 JavaBeans 组件 293
15.1 JavaBeans 组件设计约定 293
15.2 为什么使用 JavaBeans 组件 294
15.3 创建和使用 JavaBeans 组件 294
15.4 设置 JavaBeans 组件属性 295
15.5 获取 JavaBeans 组件属性 297

第 16 章 JSP 页面中的自定义标签 299
16.1 什么是自定义标签 299
16.2 JSP 页面示例 300
16.3 使用标签 301
16.4 定义标签 304
16.5 示例 314

第 17 章 JSP 标准标签库 323
17.1 JSP 页面示例 323
17.2 使用 JSTL 324
17.3 表达式语言支持 326
17.4 核心标签 328
17.5 XML 标签 331
17.6 国际化标签 333
17.7 SQL 标签 334

第 18 章 xrpcc 工具 337
18.1 语法 337
18.2 配置文件 338

第 19 章 HTTP 概述 341
19.1 HTTP 请求 341
19.2 HTTP 响应 341

附录 Java 编码方案 343

第1章　Web服务介绍

本章要点：

- XML和Java平台的角色
- 什么是XML
- Java API for XML概述
- JAXP
- JAX-RPC
- JAXM
- JAXR
- 示例

顾名思义，Web服务是通过Web网络所提供的服务。在一个典型的Web服务情形中，业务应用程序使用位于HTTP之上的SOAP协议给位于某个URL的服务发送一个请求。服务接收到请求，对它进行处理并返回一个响应。一个经常提到的Web服务的例子是股票报价服务。在这种服务中，客户请求某支股票的当前价格，然后服务器用股票的价格作为响应。这是最简单的一种Web服务形式，因为请求几乎是立即得到了满足，而且请求和响应都是同一方法调用的一部分。

另外一种Web服务的例子则是为货物的交付制订一条高效率的运输路线。在这种情况下，业务发送一个请求，其中包括货物交付的目的地，服务对该请求进行处理，确定一条成本-效益最高的交付路线。从发出请求到返回响应所花费的时间与路线的复杂程度有关，但是响应将作为一个与请求不同的操作而被发送出去。

Web服务和其客户之间通常都是业务（business）关系，这使得Web服务几乎无一例外地成为了业务-业务（B-to-B，business-to-business）事务处理过程。某个企业可以是Web服务的提供者，同时又是其客户。例如，一个调味品批发商在其使用某个Web服务来检查香草豆是否有货时处于客户角色，而当它将来自不同供应商的香草豆的价格提供给预期的客户时，它又扮演了提供者的角色。

1.1　XML和Java平台的角色

Web服务依赖于当事方相互进行通信的能力，即使它们使用了不同的信息系统和不同的数据格式。作为一种使数据可以移植的标记语言，XML是满足Web服务上述需求的关键技术。越来越多的企业发现了使用XML进行数据集成的好处，它既可以在内部的部门之间共享原有的数据，又可以对外与其他的企业进行数据共享。因此，不管是在紧耦合还是松散耦合的系统中，XML正日益被用于企业集成应用。由于XML的数据集成能力，XML正在成为Web相关计算的基础。

Web服务同时还依赖于企业使用不同计算平台来彼此进行通信的能力。这一需求使得Java平台（支持代码的可移植性）很自然地成为了开发Web服务的选择。随着新的Java API for XML的出现（它使得从Java编程语言使用XML变得更加简单），这一选择更加富有吸引力。在本章后面将总结这些API，并且将在本书后面逐个对它们进行详细介绍。

除了数据可移植性和代码可移植性之外，Web服务还必须是可升级的、安全的和高效率的，尤其是当它们不断发展时更需要这些特性。Java 2平台企业版（J2EE，Java 2 Platform，Enterprise Edition），便是专门为了满足这些需求而设计的。它可以使得Web服务开发中最困难的部分［即基础设施（infrastructure）的编程（或称为管道工程（plumbing））］变得更容易一些。这一基础设施包括诸如安全性、分布式事务处理管理以及连接池管理等特性，所有这些对于一个具有工业级强度的Web服务来说是必需的。而且由于组件是可重用的，所以开发时间可以显著减少。

由于XML和Java平台是完美的一对组合，所以它们已经在Web服务中扮演了中心角色。实际上，由于Java API for XML和J2EE平台所具有的优点，便得它们成为针对部署Web服务的完美组合。

本教程中介绍的API是J2EE API的补充，而且所处的层次在后者之上。这些API使得Java社团、开发人员、工具和容器供应商开始使用标准的Java API来开发Web服务应用和产品，这些应用和产品继续保持了Java技术的“一次编写，到处运行”这一基本特性。Java Web Services开发包（Java WSDP，Java Web Services Developer Pack）在一个单一的程序包中提供了所有这些API。Java WSDP包含了实现这些API的JAR文件以及文档和示例。在Java WSDP中的示例可以在Tomcat容器（为了便于使用，该容器在Java WSDP中包含）中运行，同时，一旦Java WSDP JAR文件在J2EE SDK中安装了之后，这些例子也可以在某个J2EE容器中运行。在J2EE SDK 1.3.1版本中提供了有关如何在J2EE SDK中安装JAR文件的指导。

本章剩余的部分将给出XML的一个概述并解释它是如何使数据成为可移植的；然后将给出Java API for XML的一个概述，描述它们所完成的工作，以及它们如何使编写Web应用程序变得更为容易。本章还分别介绍了各个API，然后提供了一个示例以说明这些API如何在一起工作。

本书随后的各章节提供了更为详细的解释并指导读者如何使用Java API for XML创建用于Web服务的应用。此外，在各章中还提供了读者可以运行的示例应用程序。

1.2 什么是XML

第2章更完整、更详细地介绍了XML。本节的目的在于使读者快速了解XML究竟是什么，它是如何使得数据成为可移植的，通过本节的介绍，可以使读者在阅读随后的Java API for XML介绍时具有相关的背景知识。

XML（Extensible Markup Language，可扩展标记语言）是一种工业标准，它以一种与系统无关的方式来表示数据。和HTML（HyperText Markup Language，超文本标记语言）一样，XML将数据封装在标签中，但是这两种标记语言之间具有一些重要的区别。首先，XML标签与被封装文本的含义具有相关性，而HTML标签只是说明了如何显示被封装的文本。以下的XML示例显示了一个价目表，其中列出了两种咖啡的名称和价格。

```
<pricelist>
    <coffee>
        <name>Mocha Java</name>
        <price>11.95</price>
    </coffee>
    <coffee>
        <name>Sumatra</name>
        <price>12.50</price>
    </coffee>
</priceList>
```

以上的 <coffee> and </coffee> 标签告诉某个解析器（parser）：在它们之间的信息是关于咖啡的。在<coffee>标签中的其他两个标签则指定了所封装的信息是咖啡的名称以及每磅的价格。由于XML标签表明了它们所封装数据的内容和结构，因此使得完成一些诸如信息存档和搜索之类的工作成为可能。

XML和HTML之间的第二个主要区别是：XML标签是可扩展的，允许写入自己的XML标签来描述数据内容。而使用HTML时，只能限于使用那些在HTML规范中预先定义的标签。

通过XML所提供的可扩展性，可以为特定类型的文档创建所需要的标签。可以使用一种XML模式语言来定义标签。模式（schema）描述了一组XML文档的结构，而且可以用于约束XML文档的内容。目前使用最广泛的模式语言可能仍然是文档类型定义（Document Type Definition）搜式语言，因为它是XML 1.0规范的一个完整部分。以这种语言所编写的模式称为一个DTD。以下的DTD定义了在价目表XML文档中使用的标签，它指定了四种标签（元素），并且进一步指定了在其他标签中可能会出现（或者需要出现）哪些标签。DTD同时还定义了XML文档的层次结构，包括标签的次序。

```
<!ELEMENT priceList (coffee)+>
<!ELEMENT coffee (name, price) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT price (#PCDATA) >
```

上面例子中的第一行给出了最高层次的元素——priceList，它表示在文档中的所有其他标签将出现在<pricelist> 和 </priceList> 标签之间。第一行还表明priceList元素必须包含一个或多个coffee元素（由加号所表明）。第二行表明了每个coffee元素必须包含一个name元素和一个price元素，而且出现的顺序如该行所示。第三行和第四行则表明位于标签<name> 和</name> 之间的数提以及位于标签<price>和</price> 之间的数据是应当进行解析的字符数据。每种咖啡的名称和价格是组成价目表的实际文本内容。

另外一种流行的模式语言是XML Schema（XML模式），它是由World Wide Web协会（W3C）开发的。XML Schema是一种功能比DTD更强大的语言。而且在2001年5月将其作为一条W3C正式建议通过之后，该语言的使用和实现日益增长。使用Java平台的开发社团已经认识到这一点，而且Java API for XML Processing（JAXP）专家组正在致力于在JAXP 1.2规范中加入针对XML Schema的支持。该版本的Java WSDP则包括了针对XML Schema的支持。

1.2.1 什么使得XML可移植

模式为XML数据提供了可移植性，前面讨论的priceList DTD就是模式的一个简单例子。如果以XML格式给某个应用程序发送一个priceList文档，同时它具有priceList DTD，则可以根据

DTD 中指定的规则来处理文档。例如，给定 priceList DTD 时，解析器将可以知道任何基于该 DTD 的文档结构和内容类型。如果该解析器是一个验证型（validating）解析器，则如果文档中包括一个在 DTD 中未包含的元素，例如元素<tea>，如果是元素没有按照预先描述的次序出现，例如 price 元素出现在 name 元素之前，则解析器将认为该文档是无效的。

XML 的其他特性也促使 XML 成为一种流行的数据交换方法。例如，XML 是以一种文本格式所编写的，这种格式是人和文本编辑软件都可读的。应用可以对 XML 文档进行解析和处理，同时人类也可以在处理过程出现错误时直接读取它们。另外一种特性是因为 XML 文档中没有包括格式化指令，它可以以多种方式显示。将数据与格式化指令分隔开意味着同一数据可以被发布到不同的媒介。

XML 支持文档的可移植性，但是它不能在真空中完成工作，换句话说，使用 XML 的当事双方必须达成特定的条件。例如，除了按照协定使用 XML 进行通信之外，两个应用之间必须协调好将使用哪些要素以及这些要素的含义。为了使用 Web 服务，这两个应用还必须协调好将使用哪些 Web 服务方法和用这些方法做什么，以及当需要一个以上方法时，这些方法的调用顺序。

企业具有一些技术可用于帮助满足以上这些需求。可以使用 DTD 和 XML 模式来描述在相互通信过程中将使用的有效名词和 XML 文档。注册表提供了一种描述 Web 服务及其方法的手段，对于更高级的模念，企业可以使用合作伙伴协议（partner agreement）和工作流程图。本书将会介绍更多有关模式和注册表的内容。

1.3 Java API for XML 概述

Java API for XML 可以使编程人员完全用 Java 编程语言编写 Web 应用程序。它们属于两大类：那些直接与处理 XML 文档打交道的 API，以及那些与程序打交道的 API。

- 面向文档的 API
  - Java API for XML Processing（JAXP）——使用各种解析器处理 XML 文档。
- 面向程序的 API
  - Java API for XML Messaging（JAXM）——在因特网上以一种标准方式发送 SOAP 消息。
  - Java API for XML Registry（JAXR）——提供一种标准方式以访问业务注册表并共享信息。
  - 基于 Java API for XML 的 RPC（JAX-RPC）——在因特网上给远程通信方发送 SOAP 方法调用并接收返回的结果。

Java API for XML 最重要的特性可能是它们都支持工业标准，这样可以确保具有互操作性。各种网络互操作性标准组织［例如 WWW 协会（W3C）以及高级结构化信息标准组织（OASIS，Organization for the Advancement of Structured Information Standards）］定义了一些操作标准，遵循这些标准的业务可以使它们的数据和应用程序协同工作。

Java API for XML 的另一个特性是它们允许各种各样的灵活性。用户可以灵活地使用 API。例如，JAXP 代码可以使用各种工具来处理一个 XML 文档，而且 JAXM 代码可以在 SOAP 之上

使用各种通信协议。同样，开发代码的人员也具有灵活性。Java API for XML定义了严格的兼容性标准以确保所有的代码实现都能够交付标准的功能，但是它也为开发人员提供了足够的自由以提供满足特定用途的实现。

下面一节将讨论各种API，从而为读者提供一个概述。

1.4 JAXP

Java API for XML Processing（JAXP）使得使用那些以Java编程语言所编写的应用程序来处理XML数据变得很容易。JAXP支持解析器标准SAX（Simple API for XML Parsing）和DOM（Document Object Model，文档对象模型），因此可以选择作为一个事件流来对数据进行解析，或者是选择创建数据的一个对象表示。JAXP的最新版本还支持XSLT（XML Stylesheet Language Transformations，XML样式表语言转换）标准，该标准提供了对数据表示的控制，并且可以将数据转换为其他XML文档或其他格式，例如HTML。JAXP还提供了名字空间支持，使得能够使用那些本来可能会产生命名冲突的XML Schema。

JAXP被设计为具有灵活性，它允许在应用中使用任何与XML兼容的解析器。这一点是通过使用称为可插层（pluggability layer）的技术来实现的。可插层允许插入一个SAX或DOM API的实现。它还允许插入一个XSL处理程序以控制XML数据的显示方式。

JAXP的最新版本为JAXP 1.2，它是一个加入了XML Schema支持的维护版本（maintenance release）。该版本当前正通过Java Community Process（JSR-63）做最后的定稿工作。在该Java WSDP发布版本中包括了一个JAXP 1.2的先期访问版本（access version），此外Java XML Pack中也提供了该版本。

1.4.1 SAX API

SAX（Simple API for XML）定义了一个用于基于事件的解析器的API。基于事件意味着解析器从头到尾阅读一个XML文档，并且每次识别一个语法结构时，它将通知运行它的应用程序。SAX解析器通过从ContentHandler接口调用方法来通知应用程序。例如，当解析器遇到一个小于号（<）时，它将调用startElement方法；当遇到字符数据时，将调用characters方法；当遇到一个后面跟随一个斜杠的小于号时，将调用endElement方法；等等。为举例说明，我们来看看示例XML文档的一部分并依次说明解析器针对每行所做的处理（为了简化起见，未包括对方法ignorableWhiteSpace的调用）。

```
<priceList>[parser calls startElement]
    <coffee>    [parser calls startElement]
        <name>Mocha Java</name>      [parser calls startElement,
                                     characters, and endElement]

    <price>11.95</price>       [parser calls startElement,
                               characters, and endElement]
</coffee>       [parser calls endElement]
```

解析器调用方法的默认实现是不完成任何操作的，因此需要编写一个子类以实现相应的方法，从而得到所希望的功能。例如，假设希望获得每磅Mocha Java的价格，则需要编写一个类