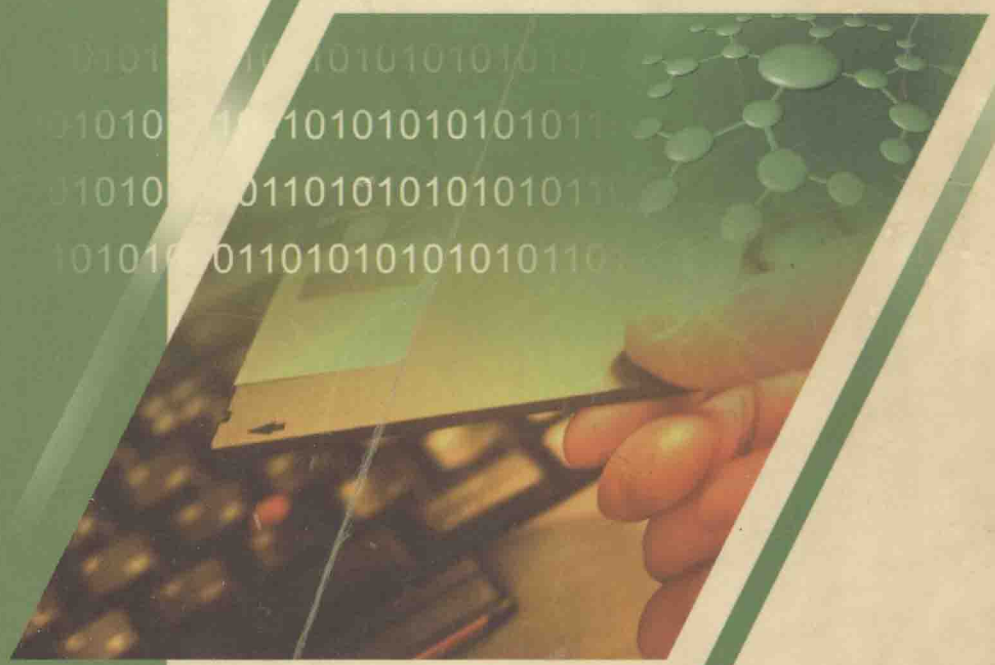
北京科海培训中心

SAMS

JSP

应用程序开发指南

[美] Ben Forta, et al. 著
章明 吴疆 译



清华大学出版社
http://www.tup.tsinghua.edu.cn

SAMS 北京科海培训中心

JSP 应用程序开发指南

[美] Ben Forta, et al. 著

章 明 吴 疆 译

清华大学出版社

(京)新登字 158 号

著作权合同登记号:01-2001-1127

内 容 提 要

JSP 是目前最流行的功能强大的 Web 应用程序开发语言。本书从介绍 JSP 的概念、语法、标记和指令入手,通过实例讲述了在 JSP 中使用企业级 JavaBean、JNDI、JDBC 和 JavaMail 等一系列实际应用,如创建消息板、创建电子商务应用、设计网站计数器等。

本书结构清晰、内容丰富,提供的最佳范例和编程技巧实用,针对 JSP 学习者是一本很好的教材。针对有经验的开发者具有很强的实用价值。

JavaServer Pages Application Development

Copyright ©2000 by Sams

本书中文简体字版由美国 SAMS 公司授权清华大学出版社和北京科海培训中心出版。未经出版者书面允许不得以任何方式复制或抄袭本书内容。

版权所有,盗版必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得进入各书店。

书　名:JSP 应用程序开发指南

作　者:Ben Forta,et al.

出版者:清华大学出版社(北京清华大学校内,邮编 100084)

印刷者:北京门头沟胶印厂

发行者:新华书店总店北京科技发行所

开　本:787×1092 1/16　印张:17.375　字数:423 千字

版　次:2001 年 6 月第 1 版　　2001 年 6 月第 1 次印刷

印　数:0001～5000

盘　号:ISBN 7-900635-62-9

定　价:34.00 元(光盘)

前　言

JSP(Java Server Pages)是一种独特的语言,它正在迅速地改变着 Web 应用程序开发的面貌。它有很好的理由做到这一点。

现在几乎所有的人都知道 Java——Sun 微系统公司创造的具有高度可移植性的面向对象的开发语言。在很短的时间内,Java 就成长为一种强健的、可信赖的语言,并且被越来越多的开发者所接受。值得注意的是 Java 从来就不是只能用在 Internet 和 Web 中。Java 是一种开发语言(也是一种完善的开发平台),它能够被用来编写从桌面到服务器的应用程序,它也当然可以用来改善 Web 应用程序。

虽然 Java 非常强大、流行,它却并不太简单——至少不像 Microsoft 的 ASP、Allaire 的 ColdFusion 甚至 Perl 那样的脚本语言一样简单。这就是这些脚本语言仍然在 Web 编程方面占据主导地位的原因。毕竟 Web 能取得成功靠的就是 HTML 的简洁性,而用同样的简洁性来构建 Web 就是一种很自然的想法。当开发者必须在 Java 的强大功能和脚本的简洁性之间作出取舍的时候,他们通常都会选择后者。

但是 JSP 的出现改变了这一切。JSP 描述了一种新的 Java 接口,它是如此地重要以至于它就是 Java 2 企业版(J2EE)规范的一部分。JSP 综合了两方面的优点——Java 强大的功能和脚本的简洁性,它为开发者在进行 Web 应用程序开发时提供了一种新的选择。JSP 使用了一种基于 XML、基于标记的语法(很像 HTML 本身),并且在编码时采用面向网页的隐喻。JSP 同时提供了对于所有 Java 类和函数的接口,而且 JSP 代码还被编译为二进制 servlets 以便于更快地进行处理。

JSP 的确是两方面的最佳选择,它也因此吸引了全世界 Web 程序员的注意。

谁将使用这本书

这本书是用来教你如何使用 JSP,不管你以前有没有使用 Java 的经验。实际上,只要你知道一些 HTML 和基本的网页制作,你就可以读懂这本书。

如果你是一个 HTML 开发人员并在寻求改善你的网站,那么这本书就适合你。如果你有用脚本语言(例如:ASP、Perl、PHP 或者 ColdFusion)开发 Web 应用程序的经验,而且想利用 Java 的强大功能,那么这本书就适合你。最后,如果你是一位经验丰富的 Java 开发人员并且要将你的知识用于 Web 开发,那么这本书就适合你。

从 Internet 基础到 Java 基础,从 Java 基础到 JSP 语法,从与接口打交道到写出自己的标记库,从通过 JDBC 获得数据库集成到通过 JavaMail 获得电子邮件,从安全性到会话状态管理,从 EJB 使用到 LDAP 集成,你都可以在这本书里面找到,而且都已经围绕着具体代码组织成极具趣味性的材料。

怎样使用这本书

这本书被设计为面向两类读者。初学者应该从头开始，系统地学习。而经验丰富的老手将会发现目录和交叉引用使本书可作为一本非常有用的参考书。

本书内容

第1章“了解JSP”。解释什么是JSP，它与Java的联系，JSP缩写的含义，以及什么是servlets。

第2章“建立JSP页面”。教你如何创建自己的JSP页面，解释如何在代码中使用Java对象和方法。

第3章“使用脚本元素”。详细介绍JSP脚本，包括变量、表达式和脚本。

第4章“使用可用对象”。解释并介绍对象、包、范围的使用。

第5章“使用Beans”。解释JavaBeans，并介绍如何在JSP代码中使用JavaBeans。

第6章“连接页面”。介绍了URL和页面链接，以及如何在页面间传递数据。

第7章“使用表单”。进一步解释HTML表单并介绍如何在JSP中使用，以及动态生成表单和表单元素。

第8章“与数据库交互”。介绍JDBC数据访问模型，它能使在你的代码内集成数据库更加方便。

第9章“保护你的应用程序”。安全性是Web应用程序中重要的一环，本章介绍安全性的实现和用户验证等方面。

第10章“管理会话状态”。介绍会话状态的管理以及如何正确使用它。

第11章“集成Email”。介绍JavaMail库，并解释如何让你的应用程序提供POP、IMAP和SMTP支持。

第12章“开发定制标记”。介绍JSP支持的新的标记库并详细解释如何创建和使用标记库。

第13章“使用企业级JavaBean”。介绍EJB的基本原理以及如何在你的代码中实现EJB。

第14章“错误处理”。介绍如何捕获并处理错误，掌握它可以让你的应用程序更加稳定，更加专业化。

第15章“调试和排错”。介绍可能出现的问题以及如何处理问题。

附录A“JSP语法”。给出JSP标记的语法及其用法的参考。

附录B“常见的JSP服务器”。列出主要的JSP服务器 及其简要说明。

附录C“使用Java方法获取CGI环境变量”。列出你可以用来获取CGI信息的Java方法。

注意：在开始深入的学习之前，你应当安装并运行一个JSP服务器，可以参考附录B“常见的JSP服务器”，同时参见本书附带的光盘。

试读结束，需要全本PDF请购买 www.ertongbook.com

目　录

试读结束，需要全本PDF请购买 www.ertongbook.com

第1章 了解JSP

这本书可以达到一个综合的目的：一方面它打算满足那些希望了解最新的Web开发和服务器端Java技术的老练的Web开发者；另一方面它又打算合乎那些刚开始接触Web开发的读者的口味。JSP(JavaServer Pages)给Web发展领域提供了一个很好的工具。了解一些基本的Web开发技术，比如：JavaScript、HTML以及CSS，会给学习和应用JSP带来不少的好处。JSP能够像HTML一样简单，也可以具有极高的技术性和挑战性。当然，它也可以介于这两者之间。

在这一章中，我们将快速回顾一下Internet和WWW的基本知识以及Web的发展情况。接下去我们会讨论一下Java——JSP的基础。如果你对Java编程语言感兴趣，那么学习JSP将是一个学习它的很好的方法；如果你对Java不感兴趣，那么可以跳过这一段，因为即使你不懂Java，一样可以用JSP做很多工作。在不久的将来，后一点可能会变得越来越真实。

1.1 Internet和WWW基础

在了解WWW的历史时要搞清楚的一个重要问题是Internet和Web间的区别。虽然很多人通常将它们混为一谈，但是它们并不是一个概念，甚至存在的时间都不一样。了解这一点对一个Web开发新手是有好处的。如果你已经清楚了这一点，那么请您稍微容忍一下我们的唠叨。

1.1.1 关于Internet

Intenet源于20世纪60年代后期的美国国防部主持下的一个小型电脑Web。它最开始的名字叫：ARPANET，Advanced Research Projects Agency Network的缩写。在接下去的岁月里，它逐渐包括了美国的各个大学和研究所，然后扩展到美国之外的大学和研究所。在20世纪70年代，Email、telnet、Usenet新闻组、Ethernet、FTP以及其他大量的Internet工具被发明和发展。许多公司也组建了内部Web。与此同时，美国政府和电信公司开始将ARPANET和其他的一些Web连接起来。到1990年时，TCP/IP、NNTP、DNS和IRC都已经发明出来了。但是从20世纪80年代早期开始到那时，Internet仍然主要是军人、科学家、大学生、研究人员和电脑迷的世界。

1.1.2 关于WWW

在20世纪90年代早期，瑞士CERN物理研究中心的一位科学家Tim Berners-Lee开始开发一种软件以可视化地反映在Internet上传输的文件。他在开发这个图像工具的过程中促进了两个相关技术的发展。其中一个是超文本传输协议(HTTP)，它是一个构建在现有的TCP/IP协议(从20世纪80年代初开始就是Internet通信的基础)之上的简单协议。另一个是一种称为超文本标记语言(HTML)的从SGML(标准广义标记语言)派生出的语言。它允

许在 Internet 上以一种用户友好的方式格式化、链接、交叉引用文档和其他信息。这就是 WWW(这种软件应用现在的名字)的由来。最开始 WWW 浏览器/编辑器只能够在 NeXT 计算机上运行,直到 1993 年最初的完全测试版本才正式发行。

你可以看到我们现在知道的 WWW 最早于 1993 年才出现。Internet 的出现远比 Web 要早。但是有趣的是,HTML、HTTP 和 Web 浏览器一出现,WWW 就发展得像火箭一样快,而 Internet 则提供了它发展的基础。

1.1.3 Web 浏览器和 Web 服务器程序

NCSA、Illinois 大学、Urbana Champaign 是 WWW 发展的温床。NCSA 开发了一种被迅速广泛应用的叫"Mosaic"的浏览器。似乎给 Internet 加一个用户界面就能够使它更易于使用,于是 NCSA 里 Mosaic 研究组领导人 Marc Andreessen 开了一家名叫 Netscape 的公司,并开发出一种 Web 浏览器,这就是广为人知的 Netscape Navigator。

同样是在 NCSA,产生了一种最初的也是最流行的 Web 服务器程序:超文本传输协议守护程序 (httpd)。Web 服务器程序开始只是一种简单而又必需的 client/server 软件,作用是响应客户端对 HTML 文件的请求。但是很快一群网管和 Web 服务器程序开发者聚集到一起,合作对 Rod McCool 在 NCSA 开发的最初的 httpd 进行改进。这个由 Brian Behlenddorf 和 Cliff Skolnick 组建的小组称这个服务器程序为 Apache,因为这是一个听上去很酷的双关语。现在 Apache 是 Internet 上应用最广的 Web 服务器程序。Apache 和其他的一些 Web 服务器程序,比如微软的 IIS 和 Netscape 的 iPlanet,现在的功能远不止处理 Internet 上的文本传输。

Web 服务器程序通常进行用户验证、数据流加密(使用安全套接层(SSL))和动态内容刷新。最后的一项就是 servlets、JSP 和其他一些技术发挥作用的地方。像 Perl 和 ASP 那样的一些技术通过能与 Web 服务器程序紧密连接的插件将 Web 服务器程序和本地应用程序编程接口(API)结合起来,以扩展 Web 服务器程序的功能。另外一些技术与 Web 服务器程序的结合就要松一些,但是却会对 Web 服务器程序进行特定的操作,然后通过通用网关接口(CGI)协议返回结果。

对用户而言,所有这些处理都是透明的。绝大多数用户知道的关于 Web 服务器程序是如何运作的一点线索是通过观察浏览器中的 URL。举例来说,如果你看到. cfm 后缀的文件,那么服务器就是使用 Allaire 的 ColdFusion 应用程序服务器;如果看到的是. php 后缀,那就是 PHP 页面;如果看到. asp,那就是微软的 ASP。如果你看到 URL 里面不是 http://而是 https://,那么你现在连接的 Web 服务器程序就是使用 SSL 加密的。依此类推。静态内容仍然很流行,但是动态内容也很广泛。大多数的网站同时包含了这两者。

1.1.4 应用程序服务器软件

client/server 计算已经取得了巨大的成功。这是一种分布式的计算,以更好地利用 Internet 上的计算资源。一种"老式的"公司级的计算方式是这样的:把大量的非智能终端和主机直接连接起来。主机是笨重而又强大的计算机,它可以同时处理数千个用户的请求。最早的商用电脑基本上都是这样的主机,它们要一整个房间才装得下。现在的主机一般就像冰箱一样大,但是运算能力却比以前大为提高。主机可能不是处处都有用的,但是大量的投资

还是在它们上，而且为它们还写有大量的定制程序。绝大多数的大公司都使用主机或者小一点的中型机来存储重要的商业数据和运行操作系统。

然而，现在的商业中处处都要讲 Web 化。不管是 B2B(business-to-business)(例如：福特公司向它的汽车零件供应商订购汽车部件)，还是 B2C(business-to-consumer)(例如：你在网上从 www. books. com 购买书籍)，这些事情都需要与世界上的其他人或者组织进行数据交换和系统整合。

在客户端的 PC 和公司的主机或数据库之间有一个巨大的中间层。中间层包括一切与数据传输有关的东西。这儿经常会有一些数据处理，特别是在数据从一个巨大的数据库中传输到终端用户的过程中。原始的数据一般是很难懂的。一个在线顾客会喜欢自己的在线购物车是一个能够单击的小图标；当他单击图标的时候他会希望网上的数据以彩色表格的形式表现出来。中间层就是一些把原始数据处理成好看页面的中间系统。

同样的，当你在 www. amazon. com 上购物时，或者在线查看公共图书馆的藏书时，或者使用网上银行系统支付账单时，那些使浏览和交流看上去十分直观的用户界面实际上只是一个包装。屏幕后面的只是一些比特和字节，从用户界面中获得的数据，一般在经过一定的处理后组织在数据库中。

当 Web 服务器程序刚开始做一些提供 HTML 页面之外的工作的时候，它与各个企业系统间没有任何联系——因为没有中间层。但是很快这就改变了。Web 服务器程序成为处理 Web 用户事务的后台系统的前沿。WWW 既被用来做零售服务处理，又被用来做 B2B 事务处理。Web 服务器程序并不能执行所有的工作，因此一些扩展程序就产生了。它们用来执行中间层任务。这整个服务器端的应用程序——执行 Web 服务器和数据库之间的数据交换——被称为应用程序服务器软件。

应用程序服务器软件是一个很含糊的概念。对一些人来说，它指的是整个网站的后台程序，包括从服务器软件到中间软件到数据库的所有的不同组件和存储器。而微软出品的称为“服务器软件”的产品进一步混乱了这个问题。当你购买 Microsoft Back Office Server 的时候，你实际上得到了 Web 服务器程序(IIS)、数据库(SQL 服务器程序)、信息服务器程序(Exchange)以及其他的东西。它是一个巨大的软件包。这种情况的最大缺点就是所有的东西都属于同一个系统，来源于同一个供应商，并且是专为 API 而写的。如果你是用 Windows 操作系统，那么你就被同一个软件供应商和操作平台所局限。

对另外一些人来说，应用程序服务器软件就是指那些连接 Web 服务器和数据库及其他后台组件的复杂的中间程序。这些应用程序服务器软件比 Microsoft Back Office 要专业化。它们对准特定的市场需要，目的是连接服务器和企业，而不是提供一个完全的混入方案。这些应用软件的例子包括：ColdFision Application Server、PHP 以及任何有着 J2EE(Java 2 企业版)商标的服务器软件。

有时候应用程序服务器软件直接植入 Web 服务器软件，就像是 Web 服务器软件的扩展包一样。ColdFusion 和 PHP 就是这样的。其他的，比如 Orion 和 JRun，既能植入 Web 服务器软件，也能独立运行，直接接受客户端的请求。后一种应用程序服务器软件的最大优点是它们不会将你局限于某一个特定的服务器端解决方案提供商，而且它们都能在多个操作平台下运行。例如，ColdFusion 能在 Linux、Solaris、HP-UX 和 Windows 下运行。一个纯粹的 Java J2EE 实现能在一切有 Java 2 JVM 的平台下运行。

1.1.5 组件、容器和连接程序

Java 2 企业版(J2EE)是一种为以上两个方面构建应用程序服务器软件的纯粹 Java 结构和一套 API。在纯粹的 Java 中有足够的 API 和规范来构建即使最复杂的商业后端。但是另一方面,J2EE 技术有着差不多同样的补充性和独立性。这就是说,你可以用 J2EE 平台的一些特性扩展你的网站,同时避免超出你所需。

有两类 JSP 和 servlet 用户:

- 一类是那些要建立动态网站,可能还会与某个在线数据库连接的用户。
- 另一类是那些要建立复杂的、安全的、易于处理的用作内部或者商业用途的中间软件的用户。

在前一种情况下,用户可以把 JSP 作为一种独立的技术,一种可以用来替换已有软件(例如 PHP、ASP、CFML、PERL 等)的软件。在后一种情况下,用户必须了解 JSP 技术在更大的 J2EE 方案中的作用;JSP 在企业应用程序服务器软件中有着特殊的作用。

不管你属于哪一类用户,接下去要讲到的 J2EE 术语都会在你要进行的应用程序开发中帮你更好的分类。

组件

组件是你的应用程序的主体构件。它包括 servlets、JavaBeans、JSP、HTML 页面、企业级 JavaBeans(EJBs)以及其他类似的东西。

容器

容器是那些执行组件的应用程序。如果你的应用程序是由 JSP 和 servlet 组件构成的,那么你就需要一个具有 JSP 支持的 servlet 容器(正式名称是 servlet 引擎);如果你的应用程序是由 EJBs 构成的,那么就要一个 EJB 容器(EJB 容器和 EJB 服务器软件在技术上是有区别的,但是这一点刚在 EJB 2.0 规范中给以区分,因此到现在为止不用考虑)。

连接程序

一般说来,连接程序就是驱动程序——就是能够让你的应用程序直接和别的软件通信的软件(和使用公共的协议比如 HTTP 不同)。连接程序把中间层和企业联系起来。它们同时也把 Java 程序员对付的纯 Java API 和大多数数据库和信息服务器程序所用的 API 联系起来。Web 服务器连接程序也可以分在这一类里面。连接程序能让你写出用于一切平台的代码而不用担心你的代码怎样连接到 Oracle 或者 Sybase 或者 MQ Series 上面去。连接程序可以做大量深层的工作使之成为可能。JDBC 驱动程序就是最常见的例子。

组件、容器和连接程序在 J2EE 中有这些专门的用途。你将会发现它们在非 J2EE 领域也是很有用的。显然,J2EE 中的一切都是从某个非 Java 版的东西发展来的,所以你可以发现这些术语在 J2EE 之外还有大量的应用。

1.2 Java的优点

就像上面提到的那样，Java在开发和配置应用程序时有一些独特的优点。在这一部分我们简介Java作为编程语言和作为服务器端开发平台的作用。我们也会复习一些使用Java的人应该知道的缩写。

1.2.1 了解Java

Java是作为一种为Internet设计的、可移植的、面向对象的编程语言而开发的。开发者是Sun微系统公司。最开始它被设计为在用户设备间进行通信。但是随着WWW的兴起，这个主意被打碎了。Sun为Internet发布了一个纯Java的Web浏览器(HotJava)，它可以下载Java插件(applets)并在浏览器中运行。Netscape也发行了一个能在程序中运行Java插件的浏览器版本。于是动态Web时代就这样产生了。Applets程序是安全的(这就是说它们运行在一个软件安全箱里，不可能对用户的系统硬件造成损害)、不依赖于操作平台的、具有Web知觉的——正好是最初设计中的三个主要特点。

客户端Java插件的最大优点之一就是能进行许多多媒体要素，因为和HTML不一样，它们实际上是一些在浏览器内部运行的小程序。一个主要的缺点是太慢而且多缺陷(bug)，并且除非程序很简单(这样就没有趣了)，下载时间太长。同时，由于它的安全性限制，它的使用范围也受到了限制。

许多客户端Java的缺点现已克服。因此现在它们可以在你的计算机上做任何让它们做的事情。下载速度的提高使得下载它们的时间可以被忽略。现在Java在客户端运行要快得多，稳定得多。但是仍然有其他的挑战。JavaScript(它和Java除了名字相似没有任何关系)、style sheets和DHTML的发展给Java的存在带来了威胁。虽然现在还有人在写一些非常酷的Java插件，但是已经非常少了。所有旧的Java插件可以实现的东西，现在JavaSript和DHTML都可以做到，而且它们能直接访问浏览器API。

在主流客户端图形界面软件方面，Java从来没有比像C++这样常用的编译语言更快过。C++是另外一种面向对象的语言，只比Java早上几年。但是Java的语法有意的以C++为基础(这是为了让C++程序员在学习Java时能更快的学会)。C++被用来写一些终端用户非常熟悉的软件，比如微软的Word和Excel，Netscape Communicator及其他。所以尽管Java现在比以前快许多，但还是比像C++这样的编译语言要慢。如果你现在要写一个Java程序，那么更可能是基于移植性的考虑而不是速度。

几年前，人们开始写服务器端的Java程序。在这里速度不是一个很重要的问题，但是也用不上Java的CPU增强图像库。当客户端的Java仍然显得很慢而使用不多时，服务器端的Java应用开始起飞。于是Sun公司又一次改变了它的发展方向。

Java在服务器上的应用已经取得了空前的成功。就在客户端Java还在不断改进，期望有一天能够成为有力的竞争者的时候，Java已经席卷了服务器市场。许多API都在服务器领域脱颖而出，业绩和规模大幅上扬，整个公司的业务都建立在Java和传统技术的结合上。

1.2.2 Java 虚拟机

Java 虚拟机(Java Virtual Machine,JVM)是 Java 可移植性的关键。它是一个通常用 C 编写的软件。Java 源程序被编译成一种被称为字节码的中间二进制语言。当你执行这个字节码的时候(也就是当你执行 Java 程序的时候),字节码就被 JVM 读入并解释。现在的 JVM 多采用实时编译(JIT)来把字节码编译成与平台相关的机器语言。JIT 可以使 Java 的执行速度接近自然语言。

在源程序和字节码的层面上 Java 是与平台无关的。也就是说,只要你把 Java 程序编译了,你就可以在任何有 JVM 的平台上运行它。秘密就是:JVM 不是用 Java 写的,它与运行它的平台紧密相连。你可以下载一个纯 Java 程序并在任何 JVM 上运行它,当然,JVM 事先必须在其所在的平台上被调试和编译为自然语言。

JVM 有很多不同的版本。在 JVM 市场上有一些竞争供应商,但是不多。IBM 和 Sun(还有 Sun 的技术分部 JavaSoft)是两个最大的竞争者,都免费提供它们的 JVM。Appeal(www.jrockit.com)是一个提供良好的、易于使用的 JVM 的小公司,它的产品对 Sun 及 IBM 的粗糙的 JVM 有很强的竞争力。

对于 Java 新手来说,听到 Java 程序实际上是在一种叫做 JVM 的程序内部运行,可能会觉得奇怪。这的确有点奇怪(而且 Java 还是一种非常成功的语言,尽管和自然语言相比比较慢),但是这并不是 Java 的特性。像 Perl、Smalltalk、Python 和 Tcl/Tk 都是这样。它们都可以编译成字节码并在 JVM 上运行,虽然很巧妙但是早就经过技术检验。

1.2.3 Java 术语

下面是一些 Java 中常见的三字缩写词。

JDK、JRE 和 JVM

我们把它们三个放在一起,是因为它们是相关的,而且经常被弄混。

JDK 也叫 SDK,因为从 Java 2 开始,Sun 同时用了两个短语来表示它:Java 软件开发包(Java Software Development Kit)和 Java 开发包(Java Development Kit)。它包括软件开发时所用的工具,也包括 JRE。

JRE 是 Java 运行环境(Java Runtime Environment)。JRE 包括所有运行基本 Java 程序要用到的运行库,也包括 JVM。

就像已经讨论过的那样,JVM 是运行 Java 程序的软件。JVM 通常是几个埋藏在 JDK 或者 JRE 目录里的文件,其中一个通常是 JVM 的范例并让它在本机操作系统上运行。当你运行 JVM 时,你必须要在它上面运行一个 Java 程序。在用户的观点上来看,JVM 的名字就是“java”。举例来说,如果你的系统是 Solaris,那么 java 命令就是一个标准命令。你只要执行它,你就执行了 JVM。如果你执行了 JVM,但是却没有指定运行的 Java 程序的名字,JVM 就什么都不会做或者只给出一个用法列表。

JVM 这个术语有好几种含义。在一种情况下,它指的是你安装 JDK 或者 JRE 时装在文件系统里面的二进制文件。当人们问:“你的 JVM 是什么版本?”时,JVM 就指的这个。

另一种情况下,它指的是 JVM 正在运行这一状况。当你执行一个 Java 程序时,JVM 的

试读结束,需要全本PDF请购买 www.ertongbook.com