HZ BOOKS
PEARSON Addison Wesley

Sun公司核心技术丛书

Java脚本编程
语言、框架与模式

Scripting in Java Languages, Frameworks,and Patterns

（美）Dejan Bosanac 著
翟育明 俞黎敏 等译

- 作者是JSR223专家组成员之一，内容权威、准确可靠。
- CSDN Java 大版主最新译作。
- 讲解脚本语言的基本概念和使用方法。
- 阐述与示例并举，用语规范标准，通俗易懂。

Sun microsystems

机械工业出版社
China Machine Press

Sun公司核心技术丛书

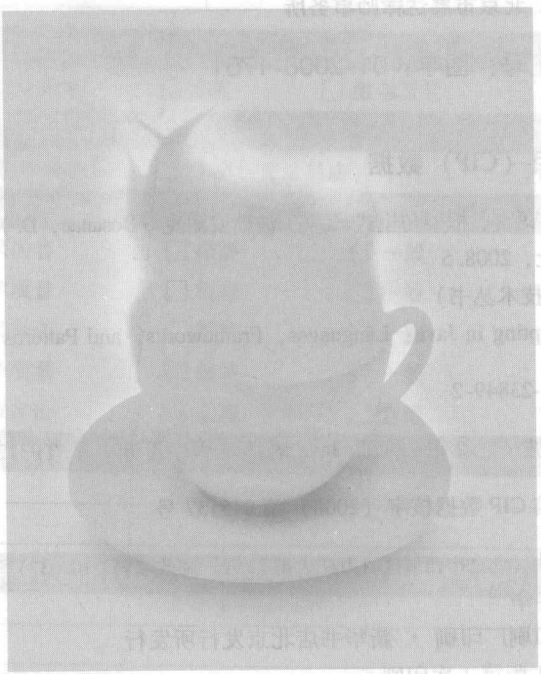
Java脚本编程

语言、框架与模式

Scripting in Java Languages, Frameworks, and Patterns

(美) Dejan Bosanac 著

翟育明 俞黎敏 等译



机械工业出版社
China Machine Press

本书讲解了脚本语言的基本概念和使用方法，概括了Java开发人员可以使用的解决方案，并探讨了在Java应用程序中应用脚本语言的用例和设计模式。内容分为五部分：第一部分介绍脚本语言的基本特征及适合用脚本语言的应用程序；第二部分介绍Java平台实际的技术和解决方案，详细讲解了Groovy脚本语言；第三部分介绍脚本语言在实际项目中的使用；第四部分介绍Java平台的脚本编程规范；第五部分提供了关于文中涉及的技术的安装和使用细节。

本书内容丰富，讲解清晰，适合作为软件开发人员的参考书。

本书版权登记号：图字：01-2008-1751

图书在版编目（CIP）数据

Java脚本编程：语言、框架与模式/（美）波斯安耐克（Bosanac, D.）著；翟育明等译. —北京：机械工业出版社，2008.5

（Sun公司核心技术丛书）

书名原文：Scripting in Java: Languages, Frameworks, and Patterns

ISBN 978-7-111-23849-2

Ⅰ. J…　Ⅱ. ①波…　②翟…　Ⅲ. Java语言－程序设计　Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第045157号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：王春华

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2008年5月第1版第1次印刷

186mm×240mm · 20.25印张

标准书号：ISBN 978-7-111-23849-2

定价：45.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线（010）68326294

译 者 序

当今，脚本语言已经在信息技术基础结构中扮演了重要的角色。它们广泛用于各种各样的任务，从工作自动化到复杂软件项目的原型和实现，而且在越来越多的应用中也显得越来越重要。随着时间的推移，脚本语言在不同的系统中有着不同的任务，但仍不可忽略传统（原生）的脚本语言在整个信息技术基础结构中的作用。

首先，本书从脚本语言的基础概念和用法入手，介绍如何通过语言的基本特征来区分脚本语言与系统编程语言，了解脚本语言幕后的概念，并且可以学到 Java 开发人员可以使用的解决方案，以及在 Java 应用程序中如何应用脚本语言和相关的设计模式。

接着，本书开始关注 Java 平台实际的技术和解决方案。目前，在 JVM 当中可以使用的有三大流行脚本语言——BeanShell、JavaScript 和 Python，在了解它们的主要特性之后，可以利用它们来与 Java 应用程序进行交互。本书通过涵盖 Groovy 内建的类似于 Java 的语法和所有的脚本概念，讨论了 Groovy 与 Java 的整合，以及一些与安全性相关的问题，并且涵盖了一些可以帮助日常编程任务的 Groovy 扩展。还介绍了如何访问数据库、创建和处理 XML 文件，以及如何利用 Groovy 中特定于脚本的特性，轻松地创建简单的 Web 应用程序和 Swing 用户界面，从而详细讲解了 Groovy 脚本语言。本书还通过 Bean Scripting Framework（BSF）讲解了一般的 Java 脚本框架。除了解释如何在项目中给任何兼容的脚本语言实现一般的支持之外，还讲解了在 BSF 中实现的基础的一些基本抽象，并介绍了一些成功的用法。

同时，本书还关注脚本语言在实际 Java 项目中的使用，关于给日常编程任务使用脚本的话题，例如，其中有单元测试、交互式调试和项目构建，Java 中的实用脚本。在脚本模式中，讲解了涉及脚本语言的 Java 应用程序设计模式。介绍了如何利用脚本来实现传统设计模式中的某些部分，并介绍了一些只特定于脚本环境的新设计模式。还讲解了这些设计模式的利与弊，以及它们的用途。

根据 JSR 223 而创建的 Scripting for the Java Platform 规范中，包括了该规范定义的两个 API：先是作为 Java 平台一个标准部分的 Scripting API，在 Mustang（Java SE 6）发行包中已经直接包含了它。Scripting API 与 BSF 一样，但是它带来了许多现代脚本框架所需要的新特性。另一个则是构建在 Scripting API 之上的 Web Scripting Framework 框架，创建它使脚本语言能够在一个 Servlet 容器内部产生 Web 内容。此外，还解释了原生的脚本语言（如 PHP）如何与 Java 平台结合，以便给 Web 应用程序开发带来更多的灵活性。

最后，在附录中提供了关于本书提及的某些技术安装和使用的细节，例如，描述了如何安装、构建和配置 Groovy 脚本语言，如何给集成开发环境安装 Groovy 支持的说明，并且还描述了如何安装第 10 章中实例运行时所需要的 JSR 223 的参考实现。

虽然我们在翻译的过程中竭力以求信、达、雅，但限于自身水平，必定会有诸多不足，还望各位读者不吝指正。大家可以通过访问我的博客 http://YuLimin. JavaEye. com/或者发送电子邮件到 YuLimin@163. com 来交流。

本书由我组织进行翻译，翟育明翻译了第 1 ~ 8 章，我翻译第 9 章和第 10 章，并负责全书所有章节的审校。参与本书翻译和审校的还有：荣浩、杨春花、凌家亮、王琳、邱庆举、李勇、师文丽、王建旭、程旭文、罗兴、刘传飞、林仪明，在此再次深表感谢。

快乐分享，实践出真知，最后，祝大家能够像我一样在阅读中享受本书带来的乐趣！

Read a bit and take it out, then come back read some more.

俞黎敏

2008 年 4 月

前　言

Java 是一种优秀的面向对象编程语言。它给软件开发人员提供了许多益处，其中包括一种很好的面向对象的方法、隐式的内存管理和动态的链接。这些语言特征就是 Java 之所以普及和广受认可的一大原因。

但是 Java 远远不只是一种编程语言，它还是一个完整的开发平台。这意味着它具备提供虚拟机的运行时环境（JRE），以及帮助开发人员完成他们想要的大多数任务的标准应用程序编程接口（API）。这种集成式运行时环境的主要好处在于它真正是平台独立的，并简化了软件开发。

另一方面，多年来，脚本语言已经在信息技术基础结构中扮演了重要的角色。它们广泛用于各种各样的任务，从工作自动化到复杂软件项目的原型和实现。

由此我们可以得出结论，Java 开发平台也可以从脚本概念和语言中受益。Java 开发人员可以在公认为最适合于这项技术的领域中使用脚本语言。就像我们将看到的，Java 平台与脚本语言的这种结合给整个软件开发过程增添了特别的品质。

本书讲解了脚本语言背后的概念，概括了 Java 开发人员可以使用的解决方案，并探讨了在 Java 应用程序中应用脚本语言的用例和设计模式。

本书编排

本书由五个部分组成。

第一部分

本书的第一部分有两章，概括描述了脚本语言：

- 第 1 章定义了脚本语言的基本特征，并将它们与系统编程语言进行对比。
- 第 2 章阐述了传统（原生）的脚本语言在整个信息技术基础结构中的作用，还讲述了随着时间的推移在不同系统中使用脚本语言的各种任务。

第二部分

讲解完脚本语言的基础概念和用法之后，我们就准备关注 Java 平台实际的技术和解决方案了。本书的这个部分包含了以下几章：

- 第 3 章从讲解 Java 平台的基础元素和解释脚本语言的适用之处开始。之后，讲解了 JVM（Java 虚拟机）可以使用的三大流行脚本语言（BeanShell、JavaScript 和 Python）的主要特性以及如何用它们来与 Java 应用程序进行交互。本章的最后讲解了 Java 开发人员可以使用的其他解决方案。

- 第4章详细讲解了Groovy脚本语言。涵盖了这种语言中内建的类似于Java的语法和所有的脚本概念，并讨论了Groovy与Java的整合，以及一些与安全性相关的问题。
- 第5章涵盖了一些有助于日常编程任务的Groovy扩展。还解释了Java程序员如何访问数据库、创建和处理XML文件，以及如何利用第4章中所涵盖的Groovy中特定于脚本的特性，轻松地创建简单的Web应用程序和Swing用户界面。
- 第6章讲解了一般的Java脚本框架。除了解释如何在工程中给任何兼容的脚本语言实现一般的支持之外，还讲解了在Bean Scripting Framework（BSF）中实现的一些基本抽象，并介绍了一些成功的用法。

第三部分

本书的这个部分主要关注脚本语言在实际Java项目中的使用：

- 第7章涵盖了用脚本完成日常编程任务的话题，例如，其中有单元测试、交互式调试和项目构建。
- 第8章讲解了涉及脚本语言的Java应用程序设计模式。介绍了如何利用脚本来实现传统设计模式中的某些部分，并介绍了一些只特定于脚本环境的新设计模式，还讲解了这些设计模式的利与弊，以及它们的用途。

第四部分

本书的这个部分涵盖了“Scripting for the Java Platform”规范，它是根据JSR（Java Specification Request）223而创建的。还特别包括了该规范定义的两个API：

- 第9章涵盖了Scripting API，这是Java平台标准的一般脚本框架。这个框架的用途与Bean Scripting Framework一样，但是Scripting API带来了许多现代脚本框架需要的新特性。Scripting API是Java平台的一个标准部分，Mustang（Java SE 6）发行包中包含了它。
- 第10章讲解了Web Scripting Framework，这个框架构建在Scripting API之上，创建这个框架来使脚本语言能够在一个Servlet容器内部产生Web内容。本章解释了原生的脚本语言（如PHP）如何与Java平台结合，以便给Web应用程序开发带来更多的灵活性。

第五部分

本书的最后由三个附录组成。这些附录的主要目的在于提供关于文中涉及的某些技术的安装和使用细节：

- 附录A描述如何安装、构建和配置Groovy脚本语言。要从文本中运行代码范例（code sample）需要有效地安装Groovy解释器。
- 附录B提供关于如何给集成开发环境（Integrated Development Environment，IDE）安装一般Groovy支持的说明。

- 附录 C 描述如何安装 JSR 223 的参考实现（Reference Implentation，RI），运行第 10 章中的实例时需要它。

我希望你会喜欢读这本书。

关于本书配套网站

本书通过网站（www. scriptinginjava. net）对内容进行了延伸，你在该网站上可以找到以下信息：

- 书中介绍的所有实例的源代码都可以下载。
- 书籍新闻、更新和增补。
- 与软件开发领域相关的新闻和信息。

致　谢

我要感谢我的家人、朋友和同事，他们在我编写本书期间给予我无限的耐心与支持。我还要感谢 Addison-Wesley 公司对本书的信任，并创造了良好的工作气氛，特别是编辑 Greg Doench 和 Ann Sellers。我还要感谢所有的技术审核人员，尤其是 George Jempty、Kevin Davis 和 Rich Rosen，他们提供了很有价值的反馈，并且在我束手无策的时候帮助我坚持方向。如果没有 Audrey Doyle，本书将晦涩难懂，感谢他帮助我编排了原稿。

最后，如果没有为书中所涵盖的项目付出辛劳的全体开发人员，本书也是不可能出版的，由衷地感谢他们。

作者简介

Dejan Bosanac 是一名专业的软件开发人员和技术顾问。他致力于各种技术的整合和互用，尤其是与 Java 和 Web 相关的应用。他花了几年的时间开发复杂的软件项目，从高流量的网站到企业级应用程序。他是 JSR 223 专家组的成员之一。

译者简介

翟育明 现任景德镇陶瓷学院国家日用及建筑陶瓷工程技术研究中心信息研究室讲师。1994 年 6 月毕业于江西师范大学计算机科学系，获理学学士学位，2006 年 2 月获华东师范大学软件工程硕士学位。曾参与国家“863”项目《陶瓷 CAD 集成设计平台》、全电办“倍增计划”项目《建筑陶瓷行业 ASP 平台》和江西省科技厅重大招标项目《面向陶瓷行业的 ASP 应用服务平台》等多项省部级科研项目。主要研究方向是电子政务、陶瓷信息化和电子商务等。

俞黎敏（ID：YuLimin，网名：阿敏总司令） 技术顾问，深入了解电力、电信行业的系统，并负责核心系统研发与管理工作。2007 年 12 月 03 日加入毕益辉系统（中国）有限公司广州分公司（BEA 广州），担任 Business Interaction Division（BID）技术顾问，主要负责 ALUI、ALBPM、PEP 产品的技术支持工作。他是开源爱好者，曾经参与 Spring 中文论坛组织“Spring 2.0 Reference”中文翻译的一审与二审工作，“满江红开放技术研究组织”的“Seam 1.2.1 Reference”中文翻译工作，并组织和完成“Seam 2.0 Reference”中文翻译工作。他利用业余时间担任 CSDN、CJSDN、Dev2DeV、Matrix、JavaWorldTW、Spring 中文等 Java 论坛版主，在各大技术社区为推动开源和敏捷开发做出了积极的贡献。博客地址为：

http://YuLimin.JavaEye.com

目 录

第三部分

第四部分

第五部分

第一部分

第1章 脚本简介

本书的主题结合了脚本技术与Java平台。它描述了Java开发人员可以用来创建一个更加强大的开发环境的项目，以及使脚本变得有用的一些实践。

在开始讨论Java领域中的脚本应用程序之前，先整体概括一下脚本背后的一些理论，以及它在信息技术基础结构中的用途。这是本书开头两章的主题，这样，可以更好地了解脚本技术，以及这项技术在Java平台内部起着什么样的作用。

首先，我们必须定义什么是脚本语言，并描述它们的特征。它们的特征很大程度上决定了它们可以（应该）用在哪些场景中。在本章中，我解释了术语脚本语言的含义，并讲解了它们的基本特征。

本章最后讨论了脚本语言与系统编程语言之间的区别，以及这些区别如何使它们符合开发中的某些场景。

1.1 背景

脚本语言的定义很含糊，有时候与脚本语言在现实中的使用方式不太一致，因此，最好整体概括一些关于编程与计算的基础概念。这样可为定义脚本语言和讨论它们的特征提供必要的基础。

让我们从头开始。处理器执行机器指令，它操作处理器的寄存器或者外部存储器中的数据。简单来说，机器指令由一系列二进制数字（很多0和1）组成，是特定于它在其中运行的特殊处理器的。机器指令由告诉处理器它应该执行什么操作的操作码和表示操作应该在其中执行的数据的操作数组成。

例如，将一个寄存器中包含的值添加到另一个寄存器中这样一个简单的操作。现在让我们假设一个简单的处理器，包含8位指令集，其中前5位表示操作码（比如，00111为“加”的寄存器值），且寄存器都以3位的模式进行处理。我们可以将这个简单的实例写成如下：

```
00111 001 010
```

在这个例子中，我用了001和010对处理器的寄存器数字1和2（相应为R1和R2）进行操作。

这个基础的计算方法已经广为人知几十年，我相信你对它也很熟悉。各种处理器对于它们指令集的外观有着不同的策略（RISC或者CISC架构），但是从软件开发人员的角度来说，唯一重要的是，处理器只能执行二进制指令。无论使用什么编程语言，最终的应用程序都是通过处理器执行的一系列机器指令。

随着时间而改变的是人们如何创建执行机器指令的顺序。这个有序的机器指令称为计算机程序。由于硬件的成本变得越来越低，且功能越来越强，因此用户的期望在增长。软件开发作为一门学科，其目标就是提供一些机制，使开发人员用与以前同样的（或者更少的）工作，就能编写出更加复杂的应用程序。

一个特定处理器的指令集称为它的*机器语言*。机器语言属于第一代编程语言。以这种方式编写的程序通常非常快，因为它们针对特殊处理器的架构进行了优化。但是除了这点好处之外，人类却很难（即便不是不可能的话）用机器语言编写大型且安全的应用程序，因为人类不善于处理大量的0和1。

为了解决这个问题，开发人员们开始为二进制数创建符号，这样就引入了*汇编语言*。汇编语言是*第二代编程语言*。汇编语言中的指令就是机器指令的上一级，它们在该级别中用容易记的关键字（如ADD、SUB等等）替代了二进制数。如此一来，你就可以用汇编语言将前面那个简单的指令实例重写成如下：

```
ADD R1, R2
```

在这个例子中，ADD关键字表示指令的操作码，R1和R2定义操作中涉及的寄存器。即使只看这个简单的实例，也可以看出汇编语言显然使程序变得更容易让人阅读，从而能够创建更复杂的应用程序。

尽管第二代语言更加面向人类了，但它们无论如何都没有扩展处理器的功能。

到了*高级语言*，它允许开发人员用更高级别的语义形式表达自己的想法，这些语言称作*第三代编程语言*。高级语言提供了各种功能强大的循环、数据结构、对象等等，使得利用它们创建许多应用程序变得更加容易。

随着时间的推移，各种不同的高级编程语言也相继出现了，它们的特征也大不相同。其中有些特征将编程语言归类为脚本（或者动态）语言，就像我们在接下来的小节中将看到的。

对于编程语言如何在主机上执行的问题上也有区别。通常，*编译器*将高级语言构造翻译成内存中的机器指令。虽然以这种方式编写的程序最初与用汇编语言编写的程序相比，效率稍微低一些，因为早期的编译器不能有效地使用系统资源，但随着时间的推移，编译器和机器都得到了改进，使得系统编程语言比汇编语言更高级。最后，高级语言在各种开发领域中变得普及起来，从业务应用程序和游戏到通信软件和操作系统实现。

但是，将高级别的语义转换成机器指令还有另外一种方法，也就是当它们被执行的时候解释它们。这样，应用程序就以它们原始的形式存在于脚本中，并且这种构造由名为*解释器*的程序在运行时进行转换。一般而言，你是在执行解释器，它读取应用程序的语句然后执行它们。这类语言称为*脚本或者动态语言*，提供了一种比系统编程语言更高级别的抽象，我们将在本章稍后详细讨论它们。

包含这些特征的语言天生就适合于某些任务，例如过程自动化、系统管理和将现有的软件组件整合在一起；简而言之，由于系统编程语言引入的严格语法和约束，在任何地方都成了开发人员与他们的工作之间的障碍。对脚本语言常用场景的描述，是第2章的重点。

但是所有这些与Java开发人员有什么关系呢？为了回答这个问题，让我们首先简要地概括一下Java平台的历史。由于平台变得越来越多样化，也为开发人员编写能在大多数现有系统中运行的软件增加了难度。就在这时候，Sun公司开发了Java语言，它具有“一次编写，随处运行”的简单性。

Java平台幕后的主要思想是将一个虚拟的处理器实现为一个软件组件，称作虚拟机。当我们有这样一台虚拟机时，就可以为该处理器编写和编译代码了，而不用特定的硬件平台或者操作系统。这个编译过程的输出称作字节码，它实际上表示目标虚拟机的机器代码。当应用程序执行时，虚拟机就启动，字节码得到解释。很显然，以这种方式开发的应用程序可以在任何安装了适当虚拟机的平台上运行。这种软件开发方法找到了许多值得关注的用处。

创建Java平台的主要动机是给客户端软件的开发创建一个良好的环境，在这个环境里，可以开发出易用、轻便、网络化的软件。但是主要由于虚拟机引起的性能损失，Java现在最适合于服务器软件开发的领域。随着个人计算机的快速增长，很显然越来越多的桌面应用程序正在用Java进行编写。这种趋势只会继续下去。

脚本语言的基本条件之一是具备一个解释器或者某种虚拟机。Java平台有Java虚拟机（JVM），它因此成为不同脚本语言之主。如今，Java社区中对这个领域的关注越来越多。已经有一些项目正在努力为Java开发人员提供传统脚本语言所拥有的功能。而且，有一种方法可以在JVM内部执行用动态语言（如Python）编写的现有应用程序，并将它与另一个Java应用程序或者模块进行整合。

这就是我们要在本书中讨论的内容。我们采用一种脚本编程方法，同时讨论这种方法的所有优缺点，包括如何在一个应用程序架构中最佳地使用脚本，以及如今在JVM内部可以用哪些工具。

1.2 脚本语言的定义

脚本语言有许多种定义，并且你看到的每一种定义都不完全符合一些典型的脚本语言。有些人根据它们的用途进行分类，有些人则根据它们的特性和引入的概念进行分类。本章讨论定义一种脚本语言的所有特征。在第2章中，我们将根据脚本语言在开发过程中的作用对它们进行分类。

1.2.1 编译器与解释器

严格说来，解释器是一种逐行执行其他高级程序的计算机程序。只被解释器执行的语言称作解释语言。

为了更好地理解编译器和解释器之间的区别，让我们简单地看一下编译器的架构（请见图1-1）。

如图1-1所示，将源代码转换成机器代码涉及几个步骤：

1. 首先，（文本形式的）源代码被逐字符地读取。扫描器将独立的字符组合成有效的语言构造（例如变量、保留字等等），称作标记（token）。

2. 标记被传到解析器，它核对程序中正在使用的语言语法是否正确。在这一步中，程序被转换成了它的解析树表示法。

3. 语义分析执行类型检查。类型检查验证源程序中已经使用的所有变量、函数等等，是否与它们的定义一致。这个解析结果是中间代码。

4. 下一步，优化器（可选地）努力生成等效的但是优化过的中间代码。

5. 在最后一步中，代码生成器从优化过的中间代码中创建目标机器代码。生成的机器代码被写成一个对象文件。