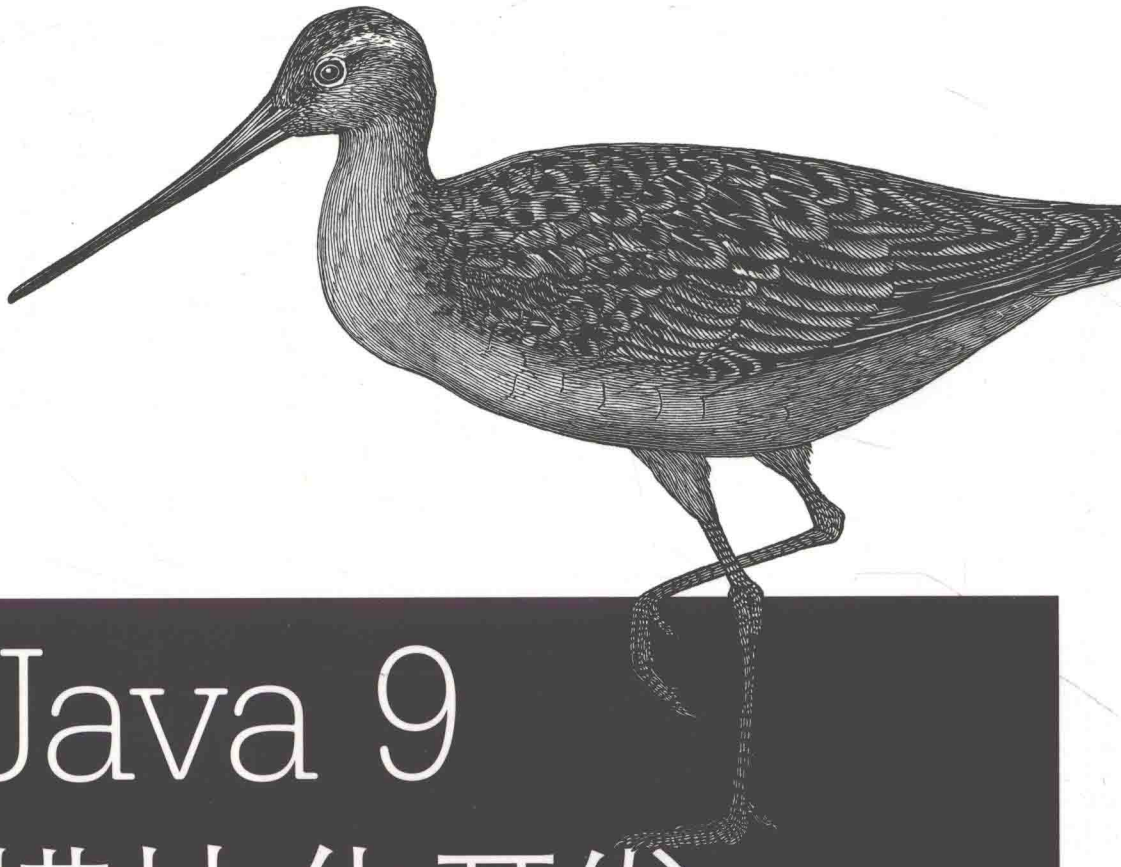
O'REILLY®

HZ BOOKS
华章IT



Java 9 模块化开发

核心原则与实践

Java 9 Modularity: Patterns and Practices for Developing Maintainable Applications

Sander Mak Paul Bakker 著
王净 等译

机械工业出版社
China Machine Press

Java 9 模块化开发：核心原则与实践

Sander Mak
Paul Bakker 著
王净 等译

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc. 授权机械工业出版社出版

机械工业出版社

图书在版编目（CIP）数据

Java 9 模块化开发：核心原则与实践 /（荷）桑德·马克（Sander Mak）等著；王净等译 . —北京：机械工业出版社，2018.5

(O'Reilly 精品图书系列)

书名原文：Java 9 Modularity: Patterns and Practices for Developing Maintainable Applications

ISBN 978-7-111-60129-6

I. J… II. ①桑… ②王… III. JAVA 语言－程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2018）第 110247 号

北京市版权局著作权合同登记

图字：01-2017-8670 号

书　　名 / Java 9 模块化开发：核心原则与实践

书　　号 / ISBN 978-7-111-60129-6

责任编辑 / 佘　洁

封面设计 / Karen Montgomery，张健

出版发行 / 机械工业出版社

地　　址 / 北京市西城区百万庄大街 22 号（邮政编码 100037）

印　　刷 / 北京诚信伟业印刷有限公司

开　　本 / 178 毫米 ×233 毫米　16 开本　15.75 印张

版　　次 / 2018 年 6 月第 1 版　2018 年 6 月第 1 次印刷

定　　价 / 69.00 元（册）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010)88379426；88361066

购书热线：(010)68326294；88379649；68995259

投稿热线：(010)88379604

读者信箱：hzit@hzbook.com

O'Reilly Media, Inc. 介绍

O'Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O'Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 Make 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar 博客有口皆碑。”

——Wired

“O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference 是聚集关键思想领袖的绝对典范。”

——CRN

“一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去 Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

译者序

JDK 9是Java开发工具包的第9个主要版本，于2017年7月下旬发布，它带来了许多令人兴奋的新功能。Java 9定义了一套全新的模块系统。当代码库越来越大，创建盘根错节的“意大利面条式代码”的概率呈指数级增长，这时候就得面对两个基础问题。首先，很难真正地对代码进行封装，而系统对不同部分（也就是JAR文件）之间的依赖关系并没有明确的概念。每一个公共类都可以被类路径之下任何其他公共类所访问，这样就会导致无意中使用了并不想被公开访问的API。其次，类路径本身也存在问题：你怎么知晓所有需要的JAR都已经有了，或者是不是会有重复的项呢？模块系统把这两个问题都解决了。

模块化的JAR文件都包含一个额外的模块描述符。在这个模块描述符中，对其他模块的依赖是通过requires来表示的。另外，exports语句控制着哪些包是可以被其他模块访问的。所有不被导出的包默认都封装在模块里。

本书共分为三部分，第一部分包括6章。第1章主要介绍了什么是模块化以及Java 9模块的主要特点。第2章学习了如何定义模块，以及使用哪些概念管理模块之间的交互。第3章在第2章的基础上通过构建自己的模块进一步学习相关模块概念。第4章讨论了可以解耦模块的服务。第5章和第6章探讨了模块化模式，以及如何以最大限度地提高可维护性和可扩展性的方式使用模块。

第二部分包括4章。第7章和第8章重点介绍了如何将现有的代码迁移到模块中。第9章通过迁移案例详细讨论了如何实现迁移。如果你是一名库的创建者或者维护者，那么第10章将对你有所帮助，其介绍了如何向库添加模块支持。

第三部分也包括4章，主要介绍了一些模块化开发工具。第11章学习了主要的IDE以及构建工具。第12章介绍了如何对模块进行测试。第13章和第14章主要介绍了自定义运行时映像以及对模块化未来的展望。

本书图文并茂、技术新、实用性强，以大量的实例对Java 9模块系统做了详细的解释，

是学习 Java 9 的读者不可缺少的实用参考书籍。本书可作为 Java 编程人员的参考手册，适合计算机技术人员使用。此外，书中还提供了相关参考资料，如果在阅读过程中遇到不明白的方法或属性，可以参阅相关内容。

参与本书翻译的人有王净、田洪、范园芳、范桢、胡训强、晏峰、余佳隽、张洁、何远燕、任方燕。最终由王净负责统稿。在此，要感谢我们的家人，他们总是无怨无悔地支持我们的一切工作。

在翻译过程中，我们尽量保持原书的特色，并对书中出现的术语和难词难句进行了仔细推敲和研究。但毕竟有少量技术是译者在自己的研究领域中不曾遇到过的，所以疏漏和争议之处在所难免，望广大读者提出宝贵意见。

最后，希望广大读者能多花些时间细细品味这本凝聚作者和译者大量心血的书籍，为将来的职业生涯奠定良好的基础。

王净

2018 年 3 月于广州

序

什么是 Java 中的模块化？对于一些人来说，模块化是一个开发原则，即对接口进行编程并隐藏实现的细节，这就是所谓的封装学派（*school of encapsulation*）。对于另外一些人来说，模块化是指依赖类加载器来提供动态执行环境，这就是所谓的隔离学派（*school of isolation*）。还有一些人认为模块化指的是工件、存储库以及相关工具，这就是所谓的配置学派（*school of configuration*）。虽然单独来看，这些观点都是正确的，但它们都太片面，感觉更像是一个不太清晰的“大故事”的几个片段。如果开发人员知道其部分代码仅供内部使用，那么他们为什么不能像隐藏类或字段一样容易地隐藏包呢？如果代码只能在依赖项存在的情况下编译和运行，那么这些依赖项为什么不能顺畅地从编译过程流向打包过程，再到安装过程，最后到执行过程呢？如果工具只有在提供了原始自描述工件时才能起作用，那么如何重用那些只是普通 JAR 文件的旧版本库呢？

Java 9 将模块作为 Java 平台的高级功能，从而很自然地引入了模块化概念。模块是一组用于重用的包，这个简单的概念对代码开发、部署以及运行的方式产生了非常深刻的影响。一旦将包放置到模块中，Java 中用来促进和控制“重用”的长期存在的机制（接口、访问控制、JAR 文件、类加载器以及动态链接）就能更好地工作。

首先，模块以其他机制无法实现的方式阐明了程序的结构。许多开发人员会惊讶地发现他们的代码结构并没有想象得那么好。例如，由多个 JAR 文件组成的代码库可以实现不同 JAR 文件中类之间的循环，但在不同模块中的类之间的循环却是禁止的。实现代码库模块化的动机之一是一旦实现了模块化，就可以避免出现因为循环依赖所产生的“泥球”（ball of mud）。[⊖]此外，使用模块进行开发还可以实现通过服务进行编程，从而减少耦合并进一步提高抽象性。

其次，模块产生了其他机制无法实现的代码责任感。从模块中导出包的开发人员实际上对 API 的稳定性做出了承诺，甚至模块本身的名称也是 API 的一部分。如果开发人员将

⊖ 泥球是指一个随意化的杂乱的结构化系统，只是代码的堆砌和拼凑，往往会导致很多错误或者缺陷。——译者注

太多的功能捆绑到单个模块中，那么就会导致该模块牵扯到大量与任何单一任务无关的依赖项；任何重用该模块的人都会意识到其杂乱无序的性质，即使模块的内部是隐藏的。使用模块进行开发可以促使每个开发人员思考其代码的稳定性和内聚性。

大多数人对桌布戏法都非常熟悉，即将桌布从桌子上迅速拿走，同时不能打翻盘子和杯子。对于那些使用 Java 9 的人来说，设计一个可以嵌入 Java 虚拟机（Java 虚拟机由自 20 世纪 90 年代以来所开发的数以百万计的类所控制）的模块系统感觉就像是反向表演桌布戏法。事实证明，模块化 JDK 导致了戏法的失败，因为一些知名的库为了自身的发展而不愿意将模块系统应用于 JDK 模块所带来的封装。Java 9 设计中的这种矛盾很难在学术上得到解决。最终，来自社区的长期反馈促使模块系统为开发人员提供了各种各样的“杠杆”和“调节盘”，使得模块化平台代码可以享受真正强大的封装，而模块化应用程序代码可以享受“足够强大”的封装。随着时间的推移，我们认为在模块化 JDK 方面进行的大胆选择将会使得代码更加可靠。

只有当一个模块系统适用于所有人时，该系统才是最好的。今天创建模块的开发人员越多，明天就会有更多的开发人员创建模块。但是那些尚未创建自己模块的开发人员又该怎么办呢？毫不夸张地说，Java 9 会像关注模块内的代码一样关注模块外的代码。代码库的作者是唯一应该对代码库进行模块化的开发人员，在完成模块化之前，模块系统必须为模块中的代码提供一种方法来“接触”模块之外的代码，而这也导致了自动模块的设计，本书将会详细介绍这部分内容。

Sander 和 Paul 都是 Java 方面的专家，同时也是 Java 9 生态系统可信任的指导者。他们身处 Java 9 开发的最前沿，是迁移流行开源库的先驱。本书面向那些对 Java 中模块化的核心原则和最佳实践感兴趣的人的，包括希望创建可维护组件的应用程序开发人员，寻求关于迁移和反射建议的库开发人员，以及希望利用模块系统高级功能的框架开发人员。我希望本书可以帮助你创建出程序结构经得起时间考验的 Java 程序。

Alex Buckley

Oracle Java 平台组

圣克拉拉，2017 年 7 月

目录

第三部分 模块化开发工具

前言

Java 9 向 Java 平台引入了模块系统，这是一个重大的飞跃，标志着 Java 平台上模块化软件开发的一个新时代的开始。看到这些变化让人感到非常兴奋，希望读者看完本书后也会感到兴奋。在深入了解模块系统之前需要做好充分利用该系统的准备。

本书读者

本书为那些想要提高应用程序的设计和结构的 Java 开发者而编写。Java 模块系统改进了设计和构建 Java 应用程序的方法。即使你不打算马上使用模块，了解 JDK 模块化本身也是非常重要的一步。在熟悉了本书第一部分所介绍的模块之后，希望你也能真正理解后续关于迁移的相关章节。

将现有代码移至 Java 9 和模块系统将成为一项越来越常见的任务。

本书绝不是对 Java 的一般性介绍。我们假设你拥有在一个团队中编写过较大 Java 应用程序的经验，在较大的 Java 应用程序中模块变得越来越重要。作为一名经验丰富的 Java 开发人员，应该认识到类路径所带来的问题，从而有助于理解模块系统及其功能。

除了模块系统之外，Java 9 中还有许多其他变化。然而，本书主要关注模块系统及其相关功能。当然，在适当的情况下，在模块系统的上下文中也会讨论其他 Java 9 功能。

编写本书的原因

很多读者从 Java 早期开始就是 Java 用户，当时 Applet 还非常流行。多年来，我们使用和喜欢过许多其他平台和语言，但 Java 仍然是主要工具。在构建可维护的软件方面，模块化是一个关键原则。多年来人们花费了大量精力来构建模块化软件，并逐渐热衷于开发模块化应用程序。曾经广泛使用诸如 OSGi 之类的技术来实现模块化，但 Java

平台本身并不支持这些技术。此外，还可通过 Java 之外的其他工具学习模块化，比如 JavaScript 的模块系统。当 Java 9 推出了期待已久的模块系统时，我们认为并不能只是使用该功能，还应该帮助其刚入职的开发人员了解模块系统。

也许在过去 10 年的某个时候你曾经听说过 Jigsaw 项目。经过多年的发展，Jigsaw 项目具备了 Java 模块系统许多功能的原型。Java 的模块系统发展断断续续。Java 7 和 Java 8 最初计划包含 Jigsaw 项目的发展结果。

随着 Java 9 的出现，长期的模块化尝试最终完成了正式模块系统的实现。多年来，各种模块系统原型的范围和功能发生了许多变化。即使你一直在密切关注该过程，也很难弄清楚最终 Java 9 模块系统真正包含什么。本书将会给出模块系统的明确概述，更重要的是将介绍模块系统能够为应用程序的设计和架构做些什么。

本书内容

本书共分为三个部分：

1）Java 模块系统介绍。

2）迁移。

3）模块化开发工具。

第一部分主要介绍如何使用模块系统。首先从介绍模块化 JDK 本身开始，然后学习创建自己的模块，随后讨论可以解耦模块的服务，最后探讨模块化模式以及如何以最大限度地提高可维护性和可扩展性的方式使用模块。

第二部分主要介绍迁移。有可能读者现在所拥有的 Java 代码不是使用专为模块系统而设计的 Java 库。该部分介绍如何将现有代码迁移到模块中，以及如何使用尚未模块化的现有库。如果你是一名库的编写者或者维护者，那么这部分中有一章专门介绍了如何向库添加模块支持。

第三部分（也是最后一部分）主要介绍工具。该部分介绍了 IDE 的现状以及构建工具。此外还会学习如何测试模块，因为模块给（单元）测试带来了一些新的挑战，也带来了机会。最后学习链接（*linking*）——模块系统另一个引人注目的功能。

虽然建议从头到尾按顺序阅读本书，但是请记住并不是所有的读者都必须这样阅读。建议至少详细阅读前四章，从而具备基本知识，以便更好地阅读本书的其他章节。如果时间有限并且有现有的代码需要迁移，那么可以在阅读完前四章后跳到本书的第二部分。一旦完成了迁移，就可以回到“更高级”的章节。

使用代码示例

本书包含了许多代码示例。所有代码示例都可以在 GitHub（https://github.com/java9-modularity/examples）上找到。在该存储库中，代码示例是按照章节组织的。在本书中，使用下面的方法引用具体的代码示例：➥ chapter3/helloworld，其含义是可以在“https://github.com/java9-modularity/examples/chapter3/helloworld”中找到示例。

强烈建议在阅读本书时使用相关的代码，因为在代码编辑器中可以更好地阅读较长的代码段。此外还建议亲自动手改写代码，如重现书中所讨论的错误。动手实践胜过读书。

排版约定

下面列出的是书中所使用的字体约定：

斜体（Italic）

表示新术语、URL、电子邮件地址、文件名以及文件扩展名。

等宽字体（`Constant width`）

用于程序清单，以及在段落中引用程序元素，如变量或函数名称、数据库、数据类型、环境变量、语句和关键字。

等宽粗体（**`Constant width bold`**）

显示应由用户逐字输入的命令或其他文本。

等宽斜体（*`Constant width italic`*）

显示应该由用户提供的值或由上下文确定的值所替换的文本。

该图标表示一般注释。

该图标表示提示或者建议。

该图标表示警告或者提醒。