

Learn Microservices with Spring Boot:
A Practical Approach to RESTful Services using RabbitMQ, Eureka, Ribbon, Zuul and Cucumber



Spring Boot 微服务实战

使用RabbitMQ、Eureka、Ribbon、
Zuul和Cucumber开发RESTful服务

[美] 莫伊塞斯·马塞罗(Moises Macero) 著
张渊 和坚 译



Spring Boot 微服务实战

使用 RabbitMQ、Eureka、Ribbon、Zuul 和
Cucumber 开发 RESTful 服务

[美] 莫伊塞斯·马塞罗(Moises Macero) 著

张渊和坚 译

清华大学出版社

北京

Moises Macero

Learn Microservices with Spring Boot: A Practical Approach to RESTful Services using RabbitMQ, Eureka, Ribbon, Zuul and Cucumber

ISBN: 978-1-4842-3164-7

Original English language edition published by Apress Media. Copyright © 2017 by Moises Macero.
Simplified Chinese-Language edition copyright © 2019 by Tsinghua University Press. All rights reserved.

本书中文简体字版由 Apress 出版公司授权清华大学出版社出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

北京市版权局著作权合同登记号 图字: 01-2019-4338

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

Spring Boot微服务实战：使用RabbitMQ、Eureka、Ribbon、Zuul和Cucumber开发RESTful服务 /
(美)莫伊塞斯·马塞罗(Moises Macero) 著；张渊, 和坚 译. —北京：清华大学出版社，2019

书名原文：Learn Microservices with Spring Boot: A Practical Approach to RESTful Services using
RabbitMQ, Eureka, Ribbon, Zuul and Cucumber

ISBN 978-7-302-53565-2

I. ①S… II. ①莫… ②张… ③和… III. ①JAVA语言—程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2019)第 179915 号

责任编辑：王军于平

装帧设计：孔祥峰

责任校对：牛艳敏

责任印制：丛怀宇

出版发行：清华大学出版社

网 址：<http://www.tup.com.cn>, <http://www.wqbook.com>

地 址：北京清华大学学研大厦 A 座 邮 编：100084

社 总 机：010-62770175 邮 购：010-62786544

投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者：三河市春园印刷有限公司

经 销：全国新华书店

开 本：170mm×240mm 印 张：15.75 字 数：317 千字

版 次：2019 年 9 月第 1 版 印 次：2019 年 9 月第 1 次印刷

定 价：59.80 元

产品编号：082810-01

译者序

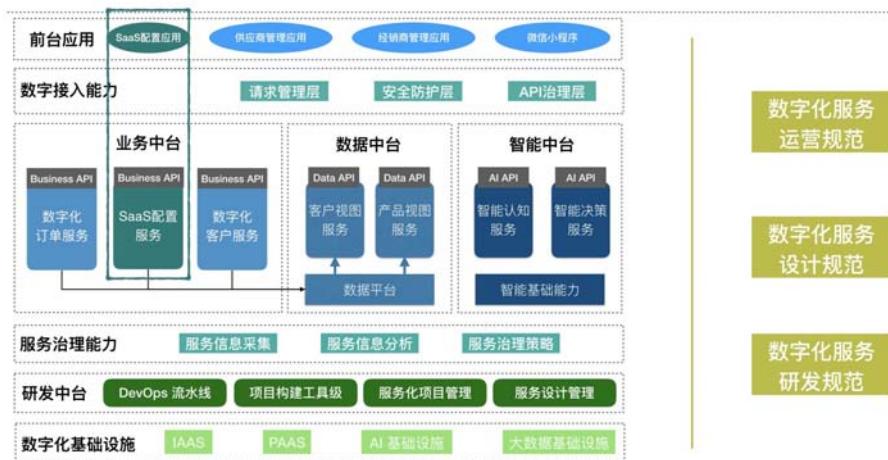
虽然市面上已经有了很多讲解 Spring Boot 的图书，但是本书绝对会给你带来不一样的体验。Moises Macero 没有把本书写得像一本使用手册一样面面俱到，也没有一上来就直接给出一个完整的微服务架构设计，然后把这个架构设计实现出来，而是像一本微服务开发者的旅行小说，带着读者一起探索微服务世界，在旅途中一起面临问题，然后一起解决问题，这样一步步用演进方式给读者呈现出最终的微服务架构设计。

这个旅途从一个业务基本需求开始，作者按照敏捷开发的方式写出了第一个用户故事并拆分出具体任务，然后使用测试驱动开发(Test-Driven Development, TDD)的方法实现了基本业务逻辑。紧接着用了一章篇幅详细介绍了一个 Spring Boot 应用的三层架构体系，并基于这个架构实现了从界面到数据库的一个完整功能。随着需求的不断增加，发现新需求不适合放在原来的应用中，于是介绍了解决这种问题的方案：多个微服务架构。实现了多个微服务以后又发现了新问题，于是开始一个个地介绍解决这些问题的工具，并把这些工具实现到系统中。最后针对复杂的微服务架构系统，发现很难进行端到端测试，因此又带着读者一起使用业务驱动开发(Business-Driven Development, BDD)的方法实现了微服务系统的端到端测试。在整个旅途中，作者在每次解决问题之前，都会深入浅出地解释为什么要这样解决。这好像在旅游时一位经验丰富的导游在给你讲解每个景点背后的故事，以及到达景点为什么要走这条路。

希望读者不仅从本书学到如何使用 Spring Boot 实现微服务，而能够通过本书更深入地理解很多微服务的底层逻辑，掌握了这套逻辑就能够使用其他开发语言或者框架实现微服务架构系统。这些年大部分公司都在把传统 IT 系统的单体架构转型成微服务架构。而基于越来越多的微服务，数字化中台这个概念开始越来越火，很多公司都提出要建设中台。构建中台的一个个微服务，每个微服务都代表企业的某种业务能力，这些业务能力通过微服务改造变成数字化服务后，企业就可以很容易地复用这些业务能力来响应业务诉求，并通过扩展微服务的方式来支持业务规模的迅速扩张。不论是微服务架构还是数字化中台，都不仅是专有的技术概念，而是一个企业在数字化时代需要具备的新技能。

下面这个图是译者提出的数字化中台全景，在本书中作者提到的服务发现、API 网关、负载均衡、服务容错等概念都能够在这个全景图中找到对应的位置。

数字化中台架构全景图



本书全部章节由张渊与和坚翻译，在这里要感谢清华大学出版社的编辑，他们为本书的翻译投入了巨大的热情并付出了很多心血。没有他们的帮助和严谨的要求，本书不可能顺利付梓。还要感谢译者的家人们，他们给予的支持保证了译者在工作之余能投入足够的时间和精力进行翻译和校对。

本书涉及了大量的实践和专业术语，译者力求翻译得准确易懂。但毕竟水平有限，错误和失误在所难免，如有任何意见和建议，请不吝指正。感激不尽！

译 者

作者简介



从孩童开始，Moises Macero 就对软件开发有浓厚的兴趣。他曾就职于一些大公司和创业公司，对于这些公司，全栈开发者是必不可少的。在 Moises 的职业生涯中，他经常为大小项目的开发、设计和架构而努力工作，包括敏捷和瀑布环境下的项目。他喜欢在不仅能指导他人，也能从他人身上学习的团队中工作。

Moises 在他的博客上分享了很多有关技术挑战的解决方案、指南及对 IT 公司工作方式的观点。在空闲时间里，他喜欢徒步旅行。

技术审校者简介



Manuel Jordan Elera 是一位自学成才的开发者和研究者，他喜欢为自己的实验学习新技术，这些实验专注于寻找新方法来集成新技术。

Manuel 获得了 2010 年的 Springy Award - Community Champion 和 2013 年的 Spring Champion。在他不多的空闲时间里，他喜欢阅读《圣经》，并用贝司和吉他作曲。Manuel 相信，持续教育和培训对于所有开发者都是必不可少的。

译者简介



张渊，ThoughtWorks 高级咨询师、软件开发人员。他曾在制造业、互联网工作，又参与了企业软件开发，使用不同的编程语言，见识着形形色色的软件架构风格，或喜或忧。他目前正在探索企业大数据实践，致力于通过工程实践提升软件质量和开发者体验。



和坚，ThoughtWorks 中台解决方案咨询师，前互联网金融公司 CTO，在企业微服务改造和中台建设方面拥有丰富的实战经验。他有十多年 IT 从业经验，从技术到金融，从金融到风控，从风控到互联网，从互联网到咨询，不断走出舒适区，体验多维的人生。

目 录

第1章 介绍	1	第3章 一个真实的三层 Spring Boot 应用	17
1.1 设置场景	1	3.1 简介	17
1.2 读者对象	2	3.2 完成基本功能	18
1.3 本书与其他图书和指南有何区别	2	3.3 领域设计	24
1.3.1 工具背后的论证	2	3.4 业务逻辑层	28
1.3.2 学习：渐进的过程	3	3.5 展示层(REST API)	30
1.3.3 这是一本指南还是一本图书	3	3.5.1 Multiplication Controller	31
1.4 本书内容	3	3.5.2 Results 控制器	35
1.4.1 从基础知识到高级话题	3	3.6 前端(Web 客户端)	38
1.4.2 搭建 Spring Boot 骨架的专业方式	4	3.7 试玩(第1部分)	42
1.4.3 测试驱动开发	4	3.8 数据持久化的新需求	43
1.4.4 连接微服务	4	3.9 重构代码	45
1.4.5 事件驱动的系统	5	3.10 数据层	50
1.4.6 端到端测试	5	3.10.1 数据模型	51
1.5 本章小结	5	3.10.2 资源库	56
第2章 一个基本的 Spring Boot 应用	7	3.11 完成第二个用户故事：串联所有层	63
2.1 业务需求	7	3.12 畅玩应用(第2部分)	69
2.2 骨架应用	8	3.13 本章小结	71
2.2.1 轻薄应用与真实应用	8		
2.2.2 创建应用骨架	8		
2.3 热身：一些 TDD 的实战	10		
2.4 本章小结	16		
第4章 初识微服务	73		
4.1 小单体之路	73		
4.1.1 单体分析	75		
4.1.2 继续前进	76		
4.2 游戏化基础	77		
4.2.1 分数、徽章和排行榜	77		

4.2.2 应用游戏化技术.....	78
4.3 转向微服务架构.....	78
4.3.1 职责分离和松耦合.....	78
4.3.2 独立变更.....	79
4.3.3 伸缩性.....	79
4.4 连接不同的微服务.....	80
4.5 事件驱动架构.....	81
4.5.1 相关技术.....	81
4.5.2 事件驱动架构的优缺点.....	82
4.5.3 深入阅读.....	84
4.5.4 应用事件驱动架构.....	84
4.6 使用 RabbitMQ 和 Spring AMQP 实现事件驱动.....	85
4.6.1 在系统中使用 RabbitMQ.....	86
4.6.2 Spring AMQP.....	86
4.7 从乘法微服务发送事件.....	87
4.7.1 RabbitMQ 配置.....	87
4.7.2 对事件建模.....	89
4.7.3 发送事件：分发器模式.....	91
4.7.4 深入新游戏化微服务.....	95
4.8 使用 RabbitMQ 接收事件.....	111
4.8.1 订阅者.....	111
4.8.2 RabbitMQ 配置.....	111
4.8.3 事件处理程序.....	114
4.9 在微服务之间请求数据.....	115
4.9.1 结合反应式模式和 REST.....	115
4.9.2 保持领域隔离.....	117
4.9.3 实现 REST 客户端.....	119
4.9.4 更新游戏化业务逻辑.....	123
4.10 使用微服务.....	126
4.11 本章小结.....	128
第 5 章 使用工具的微服务架构.....	131
5.1 介绍.....	131
5.2 抽取 UI 部分，并连接游戏化服务.....	132
5.2.1 移动静态内容.....	133
5.2.2 连接 UI 和游戏化服务.....	134
5.2.3 改变现有服务.....	136
5.2.4 全新的、更好的 UI，而且(几乎)不需要额外的代价.....	139
5.3 当前架构.....	146
5.4 服务发现和负载均衡.....	147
5.4.1 服务发现.....	147
5.4.2 负载均衡.....	149
5.4.3 多语言系统、Eureka 以及 Ribbon.....	151
5.5 通过 API 网关路由.....	153
5.5.1 API 网关模式.....	153
5.5.2 让 Zuul、Eureka 和 Ribbon 一起工作.....	156
5.6 动手准则.....	158
5.6.1 使用 Zuul 实现 API 网关.....	158
5.6.2 使用服务发现.....	172
5.6.3 微服务准备好扩展了吗.....	174
5.6.4 通过 Ribbon 实现负载均衡.....	176
5.7 断路器和 REST 客户端.....	183
5.7.1 Hystrix 断路器.....	183
5.7.2 Hystrix 和 Zuul.....	183
5.7.3 来自 REST 客户端的 Hystrix.....	186
5.7.4 使用 Feign 的 REST 消费者.....	189
5.8 微服务模式和 PaaS.....	189
5.9 本章小结.....	190

第 6 章 测试分布式系统	193		
6.1 介绍	193	6.4.7 运行测试并检查报告	225
6.2 设置场景	194	6.5 本章小结	226
6.3 Cucumber 的工作方式	195		
6.4 动手准则	197	附录 A 升级到 Spring Boot 2.0	229
6.4.1 创建一个空项目并选择工具	197	A.1 介绍	229
6.4.2 让系统可测试	200	A.2 升级依赖项	229
6.4.3 编写第一个 Cucumber 测试	207	A.3 修复已破坏的变化	232
6.4.4 把功能测试连接到 Java 代码	210	A.3.1 CrudRepository 接口 不包含 findOne() 方法	232
6.4.5 支持类	218	A.3.2 actuator 端点被移动	233
6.4.6 在 feature 之间重用步骤	223	A.4 应用可选的更新	234
		A.5 使用 Spring Boot 2.0	234
		后记	235

第1章

介 绍

1.1 设置场景

微服务现在非常流行，这并非意外。这种风格的软件架构具有许多优势，例如灵活性和易于扩展。按照微服务来组织一个个的小型团队，也能提高开发效率。然而，仅仅知道微服务的优势而去冒险是错误的：你需要理解正在面对的问题，并且未雨绸缪。你可以从互联网上的很多书籍和文章中获取大量知识，但是在着手编写代码时，就不是那么简单了。

本书通过实战方式涵盖了微服务中最重要的一些概念，并解释了相关概念。首先，我们定义了一个用例：一个需要构建的应用。然后基于合理的原因，从一个小型单体开始。一旦构建了最小的应用，我们就评估是否值得转向微服务，以及这么做的最佳方式。如何让不同的部分进行通信呢？然后描述和引入事件驱动的架构模式来达到松耦合的目的：将处理流程中发生的情况告诉系统的其他部分，而不是显式地调用其他部分来做出行动。一旦有了合适的微服务，就能看到服务发现、路由等相关工具的工作方式。我们不会一次性地介绍所有内容，但是会逐个引入到应用中，并解释它们对于应用的好处。同时，我们会分析端到端地测试分布式系统的最佳方式。

如此循序渐进的方式，会在需要时暂停下来并专注于一些概念上，这样你会理解每个工具试图解决的问题是什么。这也是为什么不断演进的示例是这本书必不可少的部分。你也可以不编写代码来理解这些概念：书中包含了许多源代码，如果你愿意也可以阅读它们。

本书中的源代码可以在 <https://github.com/microservices-practical> 上获得。它包含了若干个版本。这样你就可以很容易地看到应用是如何随着章节而进行演进的。本书包括了每个章节的对应版本的笔记。

1.2 读者对象

首先问一个问题：你对这本书有多大兴趣？这本书是实战性的，所以我们玩一个游戏。如果你认同下面的任何一条，那么本书就适合你。

- 我想学习如何使用 Spring Boot 来构建微服务，以及如何使用相关的工具。
- 每个人都在讨论微服务，在阅读了那些理论的解释和仅仅是夸大其词的文章后，我对微服务还是一无所知。即使我在 IT 行业中工作，我还是理解不了微服务的优点……
- 我想学习如何设计和开发 Spring Boot 应用，但是朋友推荐大块头的书，有时甚至推荐好几本。有没有一种来源可以让我在不阅读 1000 页书的情况下快速和实用地掌握微服务？
- 我有一个新工作，团队使用了微服务架构。我一直工作在大型单体项目上，所以我想获得一些知识和指导，来学习微服务架构的工作方式。
- 每次我进入自助餐厅时，开发人员都在讨论微服务……如果我不理解同事们在说什么，我都无法跟他们开展社交活动。好吧，这是个笑话。不要因为这个来阅读本书，尤其你对编程没有兴趣的时候。

至于阅读本书所要具备的知识，你应该熟悉以下主题。

- Java(这本书的一些代码使用了 Java 8)
- Spring(不需要有太丰富的经验，但是你至少应该理解依赖注入的工作原理)
- Maven(如果熟悉 Gradle，那也没问题)

1.3 本书与其他图书和指南有何区别

1.3.1 工具背后的论证

软件开发人员和架构师会阅读许多技术图书和指南，要么出于对新技术的兴趣，要么仅仅因为工作的缘故。这是一个不断变化的世界，我们无论如何都得学习新技术。但仅仅因为追新而去使用某种新技术是不可取的。只有理解工具背后的论证过程，才可能采用最佳的使用方式。

本书采用了这样的方式：在代码和设计模式中探索，并解释各个决策背后的原因。

1.3.2 学习：渐进的过程

如果你查看互联网上的指南，就会很快注意到它们并不是真实生活中的例子。通常，那些示例应用并不适合更复杂的场景。这些指南过于浅显，以至于对真实项目用处不大。

图书在这方面更胜一筹。很多图书围绕一个例子来解释概念，这种方法不错：因为如果看不到代码，就很难将理论概念应用到代码中。这些图书的问题是，它们比不上指南的实战性。你需要先阅读图书来理解概念，然后通过代码编写示例(或者阅读示例)，但这些示例通常都是完整的一大块。当你直接阅读最终的版本时，很难将概念带入实践。这本书偏重实战，从代码开始，通过重构进行演进。因此这些概念是一步步地理解的。在公布解决方案之前，我们先讨论问题。

由于通过渐进的方式来呈现概念，因此本书也允许你一边学习一边编码，并且独自应对挑战。

1.3.3 这是一本指南还是一本图书

本书不能被称为指南：15分钟或者30分钟不足以完成阅读。但本书也不是那种通过代码片段来呈现概念的图书。相反，在了解了可以从阅读本书的过程中获取的好处之后，你会从不是最优版本的代码开始，学习如何进行演进。

学习本书的更好方式是在阅读的同时编写代码，并且尝试书中提到的选项和其他替代方案。

无论如何，保持简单，从现在开始我们将其称作一本图书。

1.4 本书内容

1.4.1 从基础知识到高级话题

本书首先会关注一些如何使用知名的架构模式来设计和实现可用于生产环境的 Spring Boot 应用的基础。然后，通过在另一个 Spring Boot 应用中介绍第 2 部分的功能，来介绍微服务相关的工具和框架。本书还会展示如何使用端到端的集成测试来支持这样的分布式系统。

如果你已经知道了如何设计 Spring Boot 应用，那么可以快速浏览第 2 章和第 3 章，然后专注在本书的第 2 部分。在第 2 部分涵盖的话题包括服务发现、路由、事

件驱动设计、使用 Cucumber 进行测试等。然后，请注意我们在第 1 部分设定的策略——测试驱动开发，专注于最小可用产品(Minimum Viable Product, MVP)，以及单体优先。

1.4.2 搭建 Spring Boot 骨架的专业方式

首先，本书会使用 Spring Boot 指导应用的创建。这主要专注于后端部分，但你将创建一个简单的 Web 页面来演示如何通过 REST API 暴露这些后端功能。

特别要指出，我们不会仅仅通过创建一些“快捷代码”来确保 Spring Boot 已经在运行：这不是本书的目标。我们将 Spring Boot 作为工具来讲解概念，但还会使用其他技术，本书中的理念在其他技术上也适用。

你会学习如何使用常见的三层模式来设计和实现应用。你会通过动手编码来实现一个渐进式的示例。对于理解如何使用专业的方式来编写应用来说，最终的效果将会很好。

1.4.3 测试驱动开发

我们使用测试驱动开发(Test-Driven Development, TDD)的方式，将先决条件映射成技术特性(就像真实开发中那样)。有时候，TDD 在工作中并不会用到技术(有很多原因导致，不都是技术原因)。但本书会从一开始就展示它的好处：为什么在编写代码之前考虑测试用例总是一个好主意？AssertJ 和 Mockito 将帮助我们有效地构建有用的测试。

1.4.4 连接微服务

第一个微服务准备就绪后，我们会引入第二个微服务与已有的功能进行交互。从此，你将会有一个微服务架构。如果只有一个微服务，那么并没有必要去尝试理解微服务的优势。在真实场景中，分布式系统的功能会被拆分成多个服务。像往常一样，为了更有实用性，你将会看到如何通过转向微服务来满足需求。

本书不仅涵盖了拆分系统的理由，也提到了伴随这个选择的劣势。一旦做出了决策，你将会学习应该使用哪些工具来让系统作为一个整体，而不是作为多个隔离的服务来工作。这些工具包括服务发现、API 网关、负载均衡和其他的支持工具。

1.4.5 事件驱动的系统

事件驱动架构是一个不总是与微服务同时出现的概念。本书使用它的原因在于，这个模式非常适合微服务架构，你将基于一些不错的示例来做出选择。

异步的思想会引入设计代码的新方式，在编写代码时你会看到这一点。本书将使用 RabbitMQ 支持异步方式。

1.4.6 端到端测试

如果你想在项目上用专业的方式编写代码，那么需要有“发布就绪”(Production-Ready)的思维模式，所以我们会用测试来保障。我们会解释如何处理微服务中最棘手的问题：端到端测试。我们会使用 Cucumber，因为它是一个在许多项目中都能完美使用的框架，能填补业务需求和测试开发之间的空白。

1.5 本章小结

本章介绍了本书的主要目的：从简单开始，通过开发一个示例项目来增长你的知识，以此讲授微服务架构的主要内容。

我们也简单地提到了本书的主要内容：使用 Spring Boot、测试驱动开发、事件驱动系统以及基于 Cucumber 的端到端测试，将单体应用演进到微服务。

第 2 章是学习路径的第一步：一个基本的 Spring Boot 应用。

第 2 章

一个基本的 Spring Boot 应用

2.1 业务需求

软件应该有一个目标：在示例中，我们纯粹是为了学习；但无论如何，我们将给出学习软件的一个理由(一个虚构的理由)。本书会广泛使用这种面向需求的方式，以便使其更加实用。

我们想编写一个应用来鼓励用户每天训练数学技能。首先，当用户访问页面时，我们将给用户呈现两位数乘法的问题。用户输入他们的别名(一个短名字)和运算结果，他们只能使用脑力计算来得到结果。用户发送数据后，页面会呈现成功或失败的结果。

为了激发用户的兴趣，我们也会引入一些简单的游戏化技术：根据用户获得的点数进行排名。用户每天参与计算以及计算成功时能够获得这些点数。我们会在结果页面展示它们。

这就是我们即将构建的完整应用的主要想法(我们的愿景)，本书将会模仿敏捷的工作方式。在这种方式下，需求以用户故事的形式出现。尽管对敏捷很多软件开发者饱受批评，但敏捷已经在主流的 IT 公司中成为标准的方法论。事实上，如果实现得当，这种方式让团队尽早地交付可使用的软件，同时从中得到有用的反馈。

在敏捷的支持下，我们从简单开始，在此基础上进行构建。现在考虑第一个用户故事，如下所示。

用户故事 1

作为应用的用户，我想看到一个随机的乘法问题并进行在线解答(不能太简单)，以便我能够使用脑力计算，让大脑每天工作。
