- Spring Boot微服务构建和微服务无缝接入AWS云平台
- 微服务开发、设计，自动化扩缩容，微服务监控
- 依托Docker容器，在容器中部署微服务
- Docker+Marathon+Mesos集群管理

Spring微服务

Spring Microservices

(印) 拉杰什．RV ◎著
文彦峰 彭艳飞◎译

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn
[PACKT] PUBLISHING

Spring 微服务

（印）拉杰什.RV 著

文彦峰 彭艳飞 译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

Spring 是一个基于 Java 平台的应用程序框架，基于 Spring 的开发基本已经成为业界的一种规范。本书将一步一步告诉你如何使用 Spring 来开发微服务，并深度学习 Spring Boot、Spring Cloud、Docker、Mesos 和 Marathon 各个主流框架的使用方法。书中的案例都是基于最新的 Spring 框架所编，这样你会学习如何编写一个最新潮、最稳定的基于 Java 语言的系统。

本书适用于高等学校计算机及相关专业的教师、学生，以及 Spring 开发人员、架构师等技术工作人员。

图书在版编目（CIP）数据

Spring 微服务 /（印）拉杰什．RV（Rajesh RV）著；文彦峰，彭艳飞译．—北京：电子工业出版社，2018.6
书名原文：Spring Microservices
ISBN 978-7-121-34085-7

Ⅰ．①S…　Ⅱ．①拉…　②文…　③彭…　Ⅲ．①互联网络—网络服务器　Ⅳ．①TP368.5

中国版本图书馆 CIP 数据核字（2018）第 077241 号

策划编辑：董亚峰　　特约编辑：穆丽丽
责任编辑：刘小琳
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：720×1 000　1/16　印张：21.5　字数：420 千字
版　　次：2018 年 6 月第 1 版
印　　次：2018 年 6 月第 1 次印刷
定　　价：88.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：liuxl@phei.com.cn，（010）88254538。

Spring 微服务之我见

文彦峰　彭艳飞

微服务正被越来越多的组织和团体关注。微服务架构有很多好处，它通过分解巨大单体式应用为多个服务方法解决复杂性问题。在功能不变的情况下，应用被分解为多个可管理的分支或服务。每个服务都有一个 API 定义清楚的边界。微服务架构模式给采用单体式编码方式很难实现的功能提供模块化的解决方案。由此，单个服务很容易开发、理解和维护。这种架构使每个服务都可以由专门开发团队来开发，并且每个微服务都可以独立部署。

但是，微服务架构也有它的不足。微服务是一个分布式的系统架构，由此带来固有的复杂性，开发者需要了解 RPC 或者 RESTful 接口之间的消息传递内容，甚至协议。如果是单体式开发，开发者就不需要过多地关注接口，只需要关注其本身业务内容的开发即可。

在微服务架构应用中，需要更新不同服务使用的数据库，甚至是不同的数据库类型。这就需要我们了解分布式事务，或者定义一个最终一致的方法，从而对开发者提出更高的要求和挑战。

另外一个挑战在于，微服务架构模式应用的改变将会波及多个服务。例如，完成一个案例，需要修改服务 A、B、C，而 A 依赖 B，B 依赖 C。在单体式应用中，只需要改变相关模块，整合变化，然后部署就可以了。相比之下，微服务架构模式需要考虑相关改变对不同服务的影响。

测试一个基于微服务架构的应用也是很复杂的任务。例如，测试一个单体式 Web 应用的 REST API 是很容易的事情；反之，采用流行的 Spring Boot 架构，

同样的服务测试需要启动与其有关的所有服务，这样就给测试带来了一定的复杂性。

部署一个微服务应用也很复杂，而一个单体式应用只要简单地在网关后面部署各自的服务即可。相比之下，一个微服务应用一般由大批服务构成，如果它们互相依赖，就需要全部部署起来，这样才能通信并完结一个业务流程。此外，还需要一个服务发现机制，用来发现与其进行通信服务的地址。

本书由浅入深的介绍了基于 Spring Boot 的微服务开发，可以帮助读者了解 Spring Boot、Spring Cloud、Docker、Mesos 和 Marathon 的使用，详细介绍了 Spring Cloud 各种能力的实现。同时也讲述了微服务的自动化扩（缩）容，以及服务的日志记录和微服务的监控。读完本书，你能够很容易的搭建一套基于 Spring Boot 的微服务系统。

本书的翻译经历了很多困难，能够顺利完成，需要特别感谢吴疆、刘子豪、宋达彬、陈灿、叶东林、张凯旋、何文雅、陈旭泉、吴以林、颜金玉等人的大力协助。

前　言

微服务是一种架构风格和模式，通过将复杂系统分解成更小的、彼此协同工作的服务，形成大规模的商业服务。微服务是自主的、自包含的和可独立部署的服务。在当今世界，许多企业构建大型的、面向服务的企业应用程序时，都默认将微服务作为标准。

Spring 框架是多年以来开发社区流行的编程框架。Spring Boot 取消了重量级的应用容器，并提供了轻量级部署——Serverless 架构应用。Spring Cloud 结合了许多 Netflix OSS 组件，并提供了一个生态系统来运行和管理大型微服务。它提供了负载均衡、服务注册、服务监控、服务网关等能力。

然而，微服务也面临着许多挑战，特别是在大规模部署的时候，如监控、管理、分发、扩容、服务发现等。单纯使用微服务而不解决这些常见的微服务问题，会导致灾难性的后果。这本书最重要的部分是一个与技术无关的微服务能力模型，有助于解决常见的微服务挑战。

本书的目标是以务实的方式指导读者实现大规模实施响应式微服务，帮助读者深入了解 Spring Boot、Spring Cloud、Docker、Mesos 和 Marathon。读者将会理解 Spring Boot 如何通过去除重量级应用服务器，来实现自主、无须服务器的部署。读者将学习 Spring Cloud 各种能力的实现，集装箱化的 Docker、Mesos 和 Marathon 的使用，并学会抽象计算资源和控制集群范围。

我相信你会喜欢这本书的每一章节。本书通过成功构思微服务，可以为你的生意增加巨大的价值。本书通过一些案例来体现微服务的实践能力，包括一个旅游领域的研究案例。最后，你将学会如何使用 Spring 框架、Spring Boot 和 Spring

Cloud 实现微服务的体系结构。它们是经过测试的、具有强大功能的工具，可以开发和部署可扩展的微服务。在本书的帮助下，你可以使用 Spring 的最新规范来构建现代的、互联网规模的 Java 应用程序。

章节概要

第 1 章“解密微服务”，涵盖了微服务的基本概念、演变过程，以及它们与面向服务的架构的关系，并且介绍了云原生和十二要素应用程序的概念。

第 2 章“用 Spring Boot 构建微服务”，介绍了使用 Spring 框架构建 REST 和基于消息的微服务，以及如何用 Spring Boot 包装它们。另外，我们还将探索 Spring Boot 的一些核心功能。

第 3 章“微服务概念的应用”，通过详细描述开发人员在企业级微服务中所面临的挑战，来阐述微服务的实践性，并且会总结成功管理微服务生态系统所需的能力。

第 4 章“微服务的演变——一个案例的学习”，通过研究 BrownField 航空公司的微服务演变案例，向读者展示如何应用前面章节所学的微服务概念。

第 5 章“通过 Spring Cloud 对微服务进行扩（缩）容”，展示了如何使用 Spring Cloud 堆栈功能扩容之前构建的微服务。本章详细介绍了 Spring Cloud 的架构和组件，以及它们是如何整合在一起的。

第 6 章“自动化扩（缩）容微服务”，演示了使用服务网关和简单的生命周期管理器，来实现微服务的弹性化和自我管理。本章向读者展示了在现实世界中，可以使服务网关更加智能。

第 7 章“日志记录和监控微服务”，涵盖了日志在开发和监控微服务中的重要性。在这里，我们将详细介绍使用开源工具进行集中式日志记录和监控的最佳实践，以及如何将它们与 Spring 项目集成。

第 8 章“用 Docker 实现容器化微服务”，解释容器化在微服务环境中的概念。本章通过 Mesos 和 Marathon 演示了如何大规模部署微服务，来替换自定义生命周期管理器的实现方式。

第 9 章“使用 Mesos 和 Marathon 管理 Dockerized 微服务”，介绍了微服务的

自动配置和部署。本章你也将学习如何使用 Docker 容器将前面构建的微服务进行大规模部署。

准备工作

第 2 章“用 Spring Boot 构建微服务”，需要下列软件来运行代码：

- JDK 1.8
- Spring Tool Suite 3.7.2 (STS)
- Maven 3.3.1
- Spring Framework 4.2.6.RELEASE
- Spring Boot 1.3.5.RELEASE
- Spring-Boot-cli-1.3.5.RELEASE-bin.zip
- RabbitMQ 3.5.6
- FakeSMTP

第 5 章“通过 Spring Cloud 对微服务进行扩（缩）容”，除了上述软件之外还需要：

- Spring Cloud Brixton.RELEASE

在第 7 章“日志记录和监控微服务”中，我们将介绍集中式日志如何在微服务中实现，这需要以下软件堆栈：

- Elasticsearch 1.5.2
- Kibana-4.0.2-darwin-x64
- Logstash 2.1.2

第 8 章“用 Docker 实现容器化微服务”将演示如何使用 Docker 进行微服务部署，这需要以下软件组件：

- Docker version 1.10.1
- Docker Hub

第 9 章“使用 Mesos 和 Marathon 管理 Dockerized 微服务”使用 Mesos 和 Marathon 将 Dockerized 微服务部署到可自动扩展的云中。需要使用下列软件组件：

- Mesos version 0.27.1
- Docker version 1.6.2
- Marathon version 0.15.3

读者对象

本书主要针对 Spring 开发人员，他们希望构建互联网规模应用程序以满足现代业务需求。本书通过检验一些真实的用例和实践代码实例，来帮助开发人员了解究竟什么是微服务，以及当今社会微服务为什么如此重要。开发人员将懂得如何构建简单的 RESTful 服务，并将其有机地发展成为真正的企业级微服务生态系统。

对架构师而言，他们使用 Spring 框架、Spring Boot、Spring Cloud 设计强大的互联网规模微服务，并用 Docker、Mesos 与 Marathon 进行管理。当他们需要在这方面寻求帮助时，本书中的能力模型将帮助架构师设计出超出本书所讨论的工具和技术的方案。

约定

在本书中，你会发现很多区分多种不同信息的文本样式。这里是一些样式的例子及其解释。

文本中的代码、数据库表名、文件夹名称、文件名、文件扩展名、路径名、虚拟 URLs、用户输入和 Twitter 句柄等如下所示：

“下列属性可以在 application.properties 进行自定义应用程序相关信息。”

代码块设置如下：

```
<parent>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-parent</artifactId>
   <version>1.3.4.RELEASE</version>
</parent>
```

任何命令行输入或输出的写法如下：

```
$ java -jar fakeSMTP-2.0.jar
```

提示和技巧显示如下：

提示

使用 AWS Lambda，研发人员可以将应用托管到云服务平台。

下载代码文件

本书的代码文件托管在 GitHub 上，你可以通过 https://github.com/369945969/Spring-Microservices 下载代码文件。

文件下载完成后，请确保使用正确的解压缩工具：

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

目 录

第 1 章

解密微服务

微服务是用于满足现代业务需求的一种架构风格和软件开发方法。微服务不是一种发明，它更多的是之前架构风格的一种演进。

我们首先会深入探索从传统单机架构演进而来的微服务架构，之后介绍微服务的定义、概念和特性。最后，我们会分析微服务的经典案例，比较微服务和其他面向服务架构（SOA）及十二因素应用架构方法的相同点和关系。十二因素应用定义了一套在云上开发应用程序的软件设计原则。

在这一章，您会学到：

- 微服务的演进。
- 微服务架构定义和示例。
- 微服务架构的概念和特性。
- 微服务架构的经典实用案例。
- 微服务和 SOA 及十二因素应用的联系。

微服务的演进

微服务是继 SOA 之后日益流行的架构模式之一，由 DevOps 和云提供补充。这一节会详细说明，近几年现代企业中颠覆性的创新趋势和技术演进是促进微服务发展的两个因素。

微服务演进的催化剂之业务需求

在这个向数字化转型的时代，企业更多地采用技术革新来增加收益和扩大客户群。企业主要使用社交媒体、手机、云、大数据和物联网来实现颠覆性创新。利用这些技术，企业找到了快速渗透市场的新途径，这给传统 IT 交付机制带来了严重挑战。

传统开发和微服务在敏捷性、交付速度和扩展性等方面的对比如图 1-1 所示。

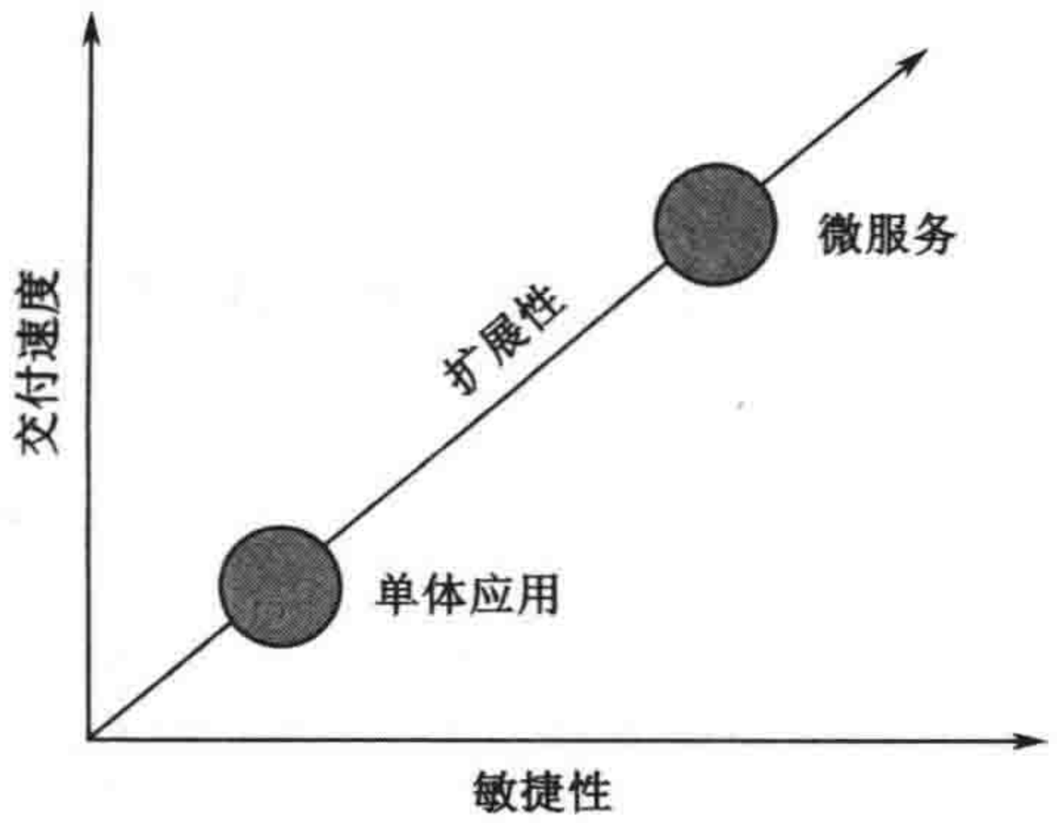


图 1-1

提示

与传统的单体应用相比，微服务更灵活，并且可以实现更快的交付速度和更大的规模部署。

在几年的转折中，商业投资大型应用程序开发的时代已经过去了。企业不再像几年前一样通过综合应用程序来管理其端到端业务功能。图 1-2 所示的成本关系图

显示了传统单体应用和微服务的周转时间和成本差异。

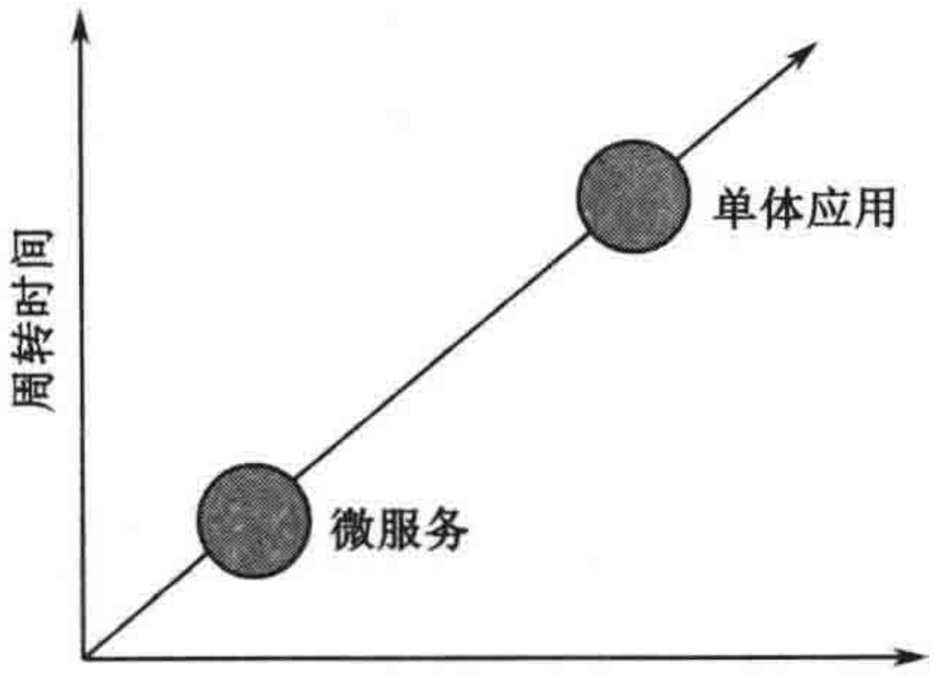


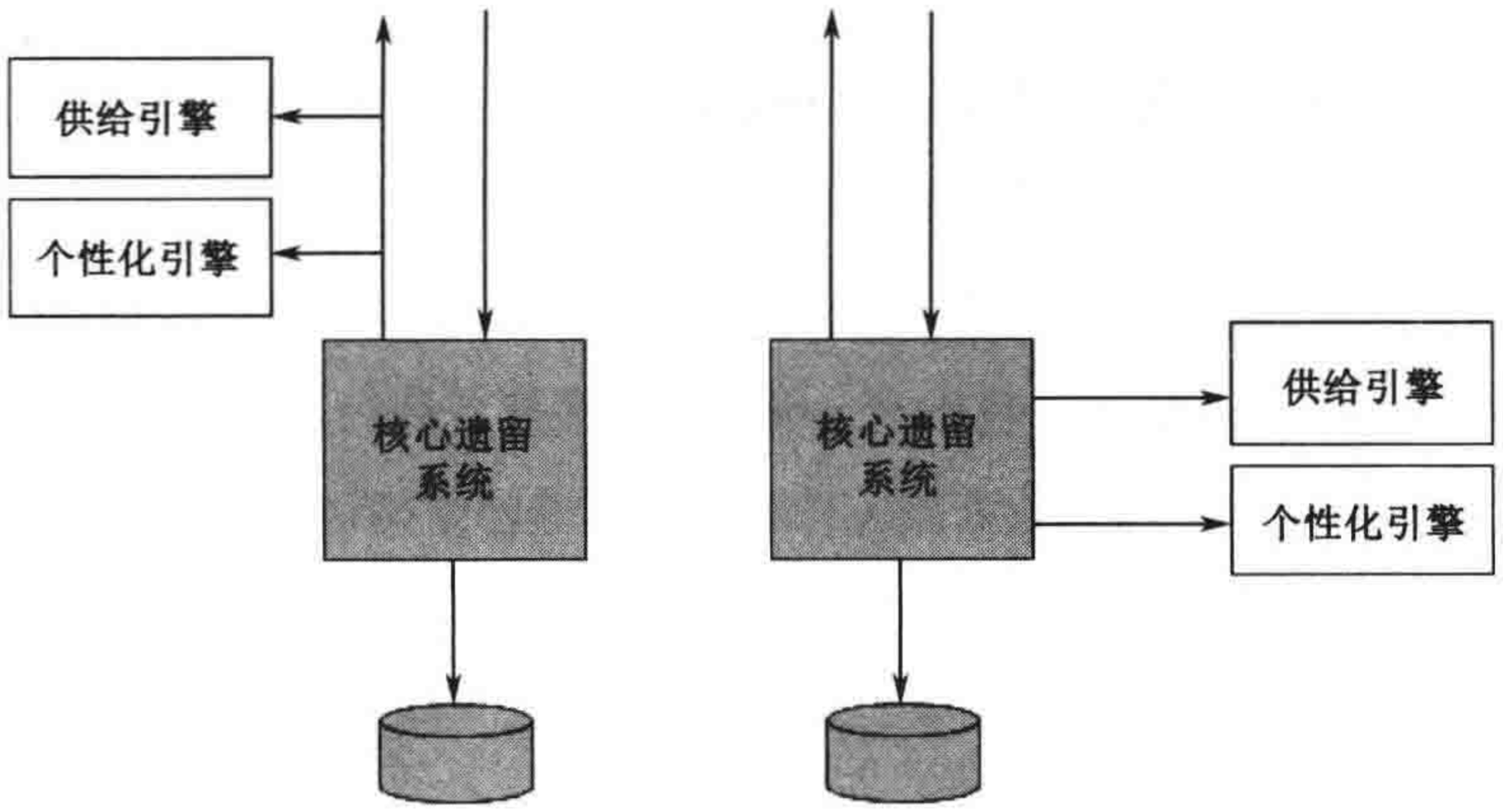
图 1-2

提示

微服务提供开发快速敏捷应用程序的方法，从而降低总体成本。

如今，航空公司和金融机构不会重建他们复杂而庞大的核心主架构系统，零售商和其他行业也不会重构重量级的供应链管理应用程序，如其传统的 ERP。焦点已转移到以最敏捷的方式来满足企业的特定需求。我们举一个使用传统的单体应用程序运行的在线零售商的例子，如果零售商想基于顾客之前的购物偏好为其提供个性化产品，或依据其购买倾向来促销产品等，他们将很快建立一个个性化引擎，或基于目前的需求为当前应用程序引入插件。

同图 1-3（a）一样，我们通常是对业务系统进行改造，进行新功能的开发，而不是去重写系统的核心功能；或如图 1-3（b）所示，修改核心系统的功能去调用一些新的功能，而这些功能通常称为微服务。



（a）通过拦截响应来引入新功能　　（b）重写核心逻辑以调用新功能

图 1-3

这种方法给团队提供了大量的机会，即可以以较低的成本快速尝试新功能。企