HZ BOOKS
华章科技

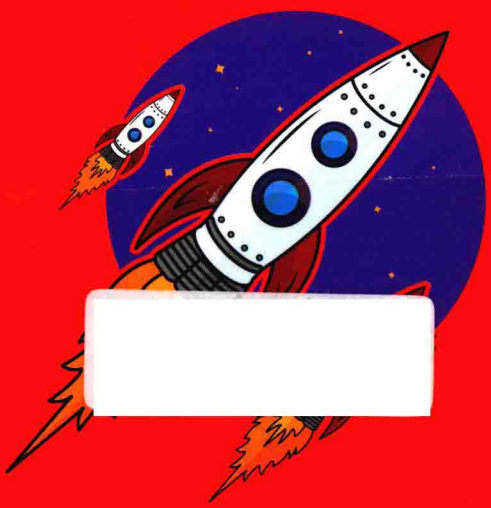
SpringBoot揭秘

快速构建微服务体系

王福强 著

阿里与平安集团技术高层倾心倾情推荐，互联网与互联网金融行业各大技术掌门一致好评。

理论与实践相结合、框架与生态相结合、技术与产品相结合，多视角、多维度、多场景地为大家深刻揭示了SpringBoot微服务框架和微服务架构体系的终极奥秘。



SpringBoot Unleashed

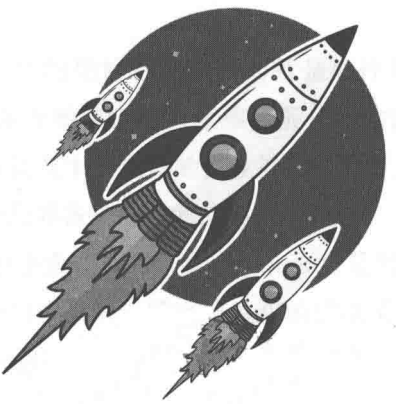
Microservices Quick Starter

机械工业出版社
China Machine Press

SpringBoot揭秘

快速构建微服务体系

王福强 著



SpringBoot Unleashed

Microservices Quick Starter

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

SpringBoot 揭秘：快速构建微服务体系 / 王福强著 . —北京：机械工业出版社，2016.5

ISBN 978-7-111-53664-2

Ⅰ. S… Ⅱ. 王… Ⅲ. 互联网络 - 网络服务器 Ⅳ. TP368.5

中国版本图书馆 CIP 数据核字（2016）第 091203 号

SpringBoot 揭秘：快速构建微服务体系

出版发行：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：李 艺 责任校对：殷 虹

印 刷：北京瑞德印刷有限公司 版 次：2016 年 5 月第 1 版第 1 次印刷

开 本：170mm × 242mm 1/16 印 张：12.5

书 号：ISBN 978-7-111-53664-2 定 价：59.00 元

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 投稿热线：（010）88379604

购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

Forword 1 推荐序 1

2015 年技术圈最火的名词大概就是微服务了。国内外的互联网技术会议上，但凡分享题目中包含“ MicroService”，不论内容质量如何，一定人山人海、摩肩接踵。

追本溯源，服务化的架构思想十年前就是软件架构的标准范式。淘宝和阿里在 2007 年左右就开始奠定了大规模服务化架构的基础，经过几代架构师的努力，有了今天承载双十一规模的商业操作系统。这中间诞生的很多优秀的 Java 中间件也成为开源界备受追崇的范例。

但是对于很多中小企业而言，SpringBoot 会是另一个性价比极高的选择。福强的这本书出现得恰逢其时，既有体系化的理论又不乏有价值的实践。对于想了解微服务和 SpringBoot 的架构师而言，是难得的修炼秘籍。

南天（本名是庄卓然） 阿里巴巴资深总监

推荐序 2 *Foreword 2*

多年前，第一次见福强，就知道他在写书，那时就是关于 Spring 的书籍。等到出书后，我翻看之下，发现福强写得非常实用。

时隔若干年，福强又来信告知有新作问世，这是他经历几年的大型网站实践之后，在创业阶段写的书。在这个阶段还能坚持写作的人非常少，足以说明他对技术的执着和坚持。有了成熟大型网站和创业阶段的实践经验，本书不仅是 SpringBoot 的指南，还是各种实战经验的提炼和总结。福强不仅在 Java，在 Scala、Golang 方面都有颇深的理解，这种跨语言方面对技术的融会贯通也为整个构建过程起着催化剂的作用。福强这次给大家带来的这本书，从不同角度对微服务这一热门话题进行了介绍和探讨，同时加入了自己多年的实践经验，值得一读。

Eric（中文名是王齐） 平安好医生 CTO

Preface 序　言

随着微服务（Micro Service）理念的盛行，一个流行的概念也随之诞生——微框架（Micro Framework），而其中最耀眼的，当属 SpringBoot。

虽然 Dropwizard 是公认的最早的微框架，但 SpringBoot “青出于蓝而胜于蓝”，背靠 Spring 框架衍生出来的整个生态体系，无论是从“出身”，还是社区的支撑上，SpringBoot 都是微框架选型的不二之选。

实际上，SpringBoot 并非单单一个微框架的概念就可以概括，笔者认为将 SpringBoot 看作一种最佳实践会更为贴切：一种 Spring 框架及其社区对“约定优先于配置”（Convention Over Configuration）理念的最佳实践。

温故而知新，笔者将通过本书带领大家回顾 Spring 框架的历史，进而引领大家探索 SpringBoot 框架的来龙去脉，最终引领大家去探索基于 SpringBoot 的微服务实践之路。希望各位能够享受这段文字旅程并有所收获。

前　言 *Preface*

为什么写这本书

忘了是2015年的哪一天，只记得几个朋友跟友商的其他几个做技术的朋友吃饭，并简单做下技术交流。席间，友商的几位朋友对SpringBoot框架实施微服务很感兴趣，交谈甚欢之际，我无意间开玩笑说："是不是该考虑写一本SpringBoot的书？"钟伦甫（原淘宝聚石）同学随口一句，"你倒是写啊！"，得，以行践言吧，谁让你把话说出去了呢？

当然，朋友的"热切期盼"只是其一，微服务盛行也是本书写作的一个契机，希望本书成为国内第一本微服务相关的原创图书，借此跟大家分享我对微服务的浅薄理解，并围绕SpringBoot微框架打造一套微服务体系可能的探索方向，权作抛砖引玉。如果不同的思想可以借此激荡和碰撞形成更多共鸣，则吾之幸甚。

因工作繁忙，只能抽取零碎时间躬耕于晨曦和月光之下，经点滴积累，才终成此书，希望大家阅读愉快。

本书的主要内容和特色

本书以介绍微服务的基本概念开篇，逐步引出Java平台下打造微服务的利器——SpringBoot微框架。书中从SpringBoot微框架的"出身"开始，循序渐进，一步步为大家剖析SpringBoot微框架的设计理念和原理，并对框架的重点

功能和模块进行了逐一讲解。

当然，这还只是“前戏”，本书最精彩的部分在于，在大家对 SpringBoot 微框架已经有了基本的认识之后，我们将一起探索如何基于 SpringBoot 微框架打造一套完备的微服务体系。因为如果没有平台化体系化的基础支撑，空谈微服务将无太大意义。

SpringBoot 微框架依托 Java 平台和 Spring 框架，拥有良好的可扩展性和可定制性，为了说明这一点，我们单独开辟了一章内容，为大家介绍如何使用 Scala 和 SpringBoot 微框架来开发和交付相应的微服务，并且围绕 Scala 和 SpringBoot 如何打造相应的工具，技术产品等支持来提高相应微服务的交付效率。

最后我会与大家一起对 SpringBoot 微框架的相关内容进行回顾和展望，以期温故而知新。

本书总体上可以总结为三个关键词，“框架、体系、生态”，三者循序渐进，相辅相成，在使用 SpringBoot 微框架打造自己特色的微服务体系和技术生态之时，希望大家记住这三个关键词。

本书面向的读者

本书希望面向的读者当然是那些对 SpringBoot 微框架感兴趣的同学，如果你想了解 SpringBoot 微框架，并且尝试进一步深入定制该框架以满足自己团队和公司的需要，也希望会对你有所启发。

除此之外还包括：

- Java 平台上的广大研发同学，可以借此书了解业界微服务相关的最新动态。
- 其他平台上的广大研发同学，可借此书“管中窥豹”，了解微服务的一般体系和生态建设，对比并引入自身的技术和微服务体系建设之中。
- 脱离技术一线已久的技术负责人。

如何阅读本书

本书采用循序渐进的形式编写，所以顺序阅读是推荐的阅读方式。

勘误和资源

鉴于一家之言且编撰仓促，难免会有所纰漏，观点有失偏颇，所以，我在github网站上专门新建了一个issue项目（https://github.com/fujohnwang/unveil-springboot-feedbacks），如果大家在阅读此书之后发现有哪些错误和疑问，或者改进建议，可以在此项目上新建issue来表达自己的观点和建议。如果时间不充裕，我会适时地选择性给予答复，当然，更希望大家可以通过issue展开讨论，互相切磋和解答疑问 。

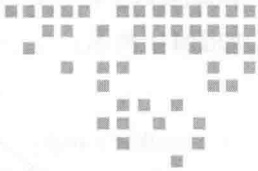
致谢

除了最初的一句戏言，钟伦甫同学也是本书的第一位读者，帮助审稿并提出很多建议，所以，本书得以出版，第一需要感谢的就是钟伦甫同学。

其次，我要感谢华章出版社的杨福川和李艺，福川兄在接到我的出版意向之后，快速地跟进和落实，在本书初稿编写完成时马上着手出版，诸位得以在2016年上半年就手捧此书，皆需感谢福川兄的重点关注和推进。

最后要感谢我的父母，感谢他们把我带到这个世界上并让我做自己想做和要做的事情。

Contents 目　录



第 1 章 Chapter 1

了解微服务

SpringBoot 是一个可使用 Java 构建微服务的微框架，所以在了解 SpringBoot 之前，我们需要先了解什么是微服务。

1.1 什么是微服务

微服务（Microservice）虽然是当下刚兴起的比较流行的新名词，但本质上来说，微服务并非什么新的概念。实际上，很多 SOA 实施成熟度比较好的企业，已经在使用和实施微服务了。只不过，它们只是在闷声发大财，并不介意是否有一个比较时髦的名词来明确表述 SOA 的这个发展演化趋势罢了。

微服务其实就是服务化思路的一种最佳实践方向，遵循 SOA 的思路，各个企业在服务化治理的道路上走的时间长了，踩的坑多了，整个软件交付链路上各个环节的基础设施逐渐成熟了，微服务自然而然就诞生了。

当然，之所以叫微服务，是与之前的服务化思路和实践相比较而来的。早些年的服务实现和实施思路是将很多功能从开发到交付都打包成一个很大的服务单元（一般称为 Monolith），而微服务实现和实施思路则更强调功能趋向单一，服务单元小型化和微型化。如果用“茶壶煮饺子”来打比方的话，原来我们是在一个茶壶里煮很多个饺子，现在（微服务化之后）则基本上是在一个茶

壶煮一个饺子，而这些饺子就是服务的功能，茶壶则是将这些服务功能打包交付的服务单元，如图 1-1 所示。

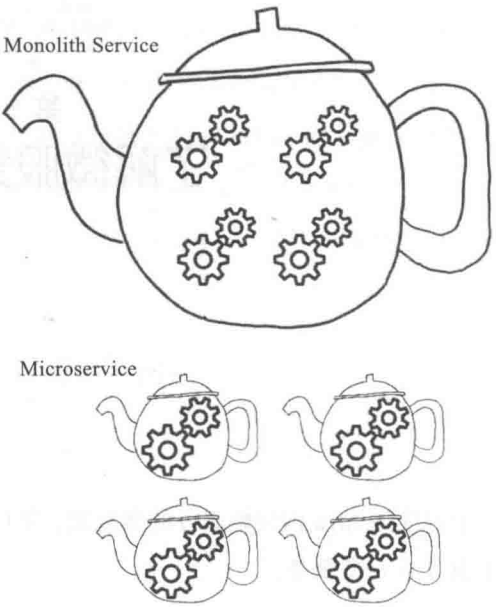


图 1-1　论茶壶里煮“饺子”的不同形式

所以，从思路和理念上来讲，微服务就是要倡导大家尽量将功能进行拆分，将服务粒度做小，使之可以独立承担对外服务的职责，沿着这个思路开发和交付的软件服务实体就叫作“微服务”，而围绕着这个思路和理念构建的一系列基础设施和指导思想，笔者将它称为“微服务体系”。

1.2　微服务因何而生

微服务的概念我们应该大体了解了，那么微服务又是怎么来的？原来将很多功能打包为一个很大的服务单元进行交付的做法不能满足需求吗？

实际上，并非原来“大一统”（Monolith）的服务化实践不能满足要求，也不是不好，只是，它有自己存在的合理场景。对于 Monolith 服务来说，如果团队不大，软件复杂度不高，那么，使用 Monolith 的形式进行服务化治理是比较合适的，而且，这种方式对运维和各种基础设施的要求也不高。

但是，随着软件系统的复杂度持续飙升，软件交付的效率要求更高，投入的人力以及各项资源越来越多，基于 Monolith 的服务化思路就开始“捉襟见肘”。

在开发阶段，如果我们遵循 Monolith 的服务化理念，通常会将所有功能的实现都统一归到一个开发项目下，但随着功能的膨胀，这些功能一定会分发给不同的研发人员进行开发，造成的后果就是，大家在提交代码的时候频繁冲突并需要解决这些冲突，单一的开发项目成为了开发期间所有人的工作瓶颈。

为了减轻这种苦恼，我们自然会将项目按照要开发的功能拆分为不同的项目，从而负责不同功能的研发人员就可以在自己的代码项目上进行开发，从而解决了大家无法在开发阶段并行开发的苦恼。

到了软件交付阶段，如果我们遵循 Monolith 的服务化理念，那么，我们一定是将所有这些开发阶段并行开发的项目集合到一起进行交付，这就涉及服务化早期实践中比较有名的“火车模型”，即交付的服务就像一辆火车，而这个服务相关的所有功能对应的项目成果，就是要装上火车车厢的一件件货物，交付的列车只有等到所有项目都开发测试完成后才可以装车出发，完成整个服务的交付。很显然，只要有一个车厢没有准备好货物（即功能项目未开发测试完成），火车就不能发车，服务就不能交付，这大大降低了服务的交付效率。如果每个功能项目可以各自独立交付，那么就不需要都等同一辆火车，各自出发就可以了。顺着这个思路，自然而然地，大家逐渐各自独立，每一个功能或者少数相近的功能作为单一项目开发完成后将作为一个独立的服务单元进行交付，从而在服务交付阶段，大家也能够并行不悖，各自演化而不受影响。

所以，随着服务和系统的复杂度逐渐飙升，为了能够在整个软件的交付链路上高效扩展，将独立的功能和服务单元进行拆分，从而形成一个一个的微服务是自然而然发生的事情。这就像打不同的战役一样，在双方兵力不多、战场复杂度不高的情况下，Monolith 的统一指挥调度方式是合适的；而一旦要打大的战役（类似于系统复杂度提升），双方一定会投入大量的兵力（软件研发团队的规模增长），如果还是在狭小甚至固定的战场上进行厮杀，显然施展不开！所以，小战役有小战役的打法，大战役有大战役的战法，而微服务实际上就是一种帮助扩展组织能力、提升团队效率的应对“大战役”的方法，它帮助我们从软件开发到交付，进而到团队和组织层面多方位进行扩展。

总的来说，一方面微服务可以帮助我们应对飙升的系统复杂度；另一个方面，微服务可以帮助我们进行更大范围的扩展，从开发阶段项目并行开发的扩展，到交付阶段并行交付的扩展，再到相应的组织结构和组织能力的扩展，皆因微服务而受惠。

1.3 微服务会带来哪些好处

显然，随着系统复杂度的提升，以及对系统扩展性的要求越来越高，微服务化是一个很好的方向，但除此之外，微服务还会给我们带来哪些好处？

1.3.1 独立，独立，还是独立

我们说微服务打响的是各自的独立战争，所以，每一个微服务都是一个小王国，这些微服务跳出了“大一统”（Monolith）王国的统治，开始从各个层面打造自己的独立能力，从而保障自己的小王国可以持续稳固的运转。

首先，在开发层面，每个微服务基本上都是各自独立的项目（project），而对应各自独立项目的研发团队基本上也是独立对应，这样的结构保证了微服务的并行研发，并且各自快速迭代，不会因为所有研发都投入一个近乎单点的项目，从而造成开发阶段的瓶颈。开发阶段的独立，保证了微服务的研发可以高效进行。

服务开发期间的形态，跟服务交付期间的形态原则上是不需要完全高度统一的，即使我们在开发的时候都是各自进行，但交付的时候还是可以一起交付，不过这不是微服务的做法。在微服务治理体系下，各个微服务交付期间也是各自独立交付的，从而使得每个微服务从开发到交付整条链路上都是独立进行，这大大加快了微服务的迭代和交付效率。

服务交付之后需要部署运行，对微服务来说，它们运行期间也是各自独立的。

微服务独立运行可以带来两个比较明显的好处，第一个就是可扩展性。我们可以快速地添加服务集群的实例，提升整个微服务集群的服务能力，而在传统 Monolith 模式下，为了能够提升服务能力，很多时候必须强化和扩展单一结点的服务能力来达成。如果单结点服务能力已经扩展到了极限，再寻求扩展的