异步图书
www.epubit.com

Packt

从实战出发，更好地理解微服务

微服务实践

Practical
Microservices

[印度] 乌姆什·拉姆·夏尔玛（Umesh Ram Sharma）著
占红来 刘博 译

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

微服务实践

Practical
Microservices

[印度] 乌姆什·拉姆·夏尔玛（Umesh Ram Sharma）著
占红来 刘博 译

人民邮电出版社
北京

图书在版编目（CIP）数据

微服务实践 / (印) 乌姆什·拉姆·夏尔玛 (Umesh Ram Sharma) 著 ; 占红来, 刘博译. -- 北京 : 人民邮电出版社, 2019.1
书名原文: Practical Microservices
ISBN 978-7-115-49870-0

Ⅰ. ①微… Ⅱ. ①乌… ②占… ③刘… 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252033号

版权声明

◆ 著　[印度]乌姆什·拉姆·夏尔玛（Umesh Ram Sharma）
译　占红来　刘　博
责任编辑　杨海玲
责任印制　焦志炜

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市祥达印刷包装有限公司印刷

◆ 开本：800×1000　1/16
印张：13
字数：242 千字　2019 年 1 月第 1 版
印数：1－2 600 册　2019 年 1 月河北第 1 次印刷

著作权合同登记号　图字：01-2017-8624 号

定价：49.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

内容提要

近些年来，微服务一直是非常热门的话题，关于微服务架构的讨论也是层出不穷。本书以贯穿整书的示例为出发点，由浅入深地阐述使用微服务的最佳实践，以及如何避免采用微服务架构可能带来的复杂性陷阱。本书从微服务架构本身的特征入手，讨论微服务组件的设计指导原则、有效通信的方式以及常见的安全挑战和数据模型的选择；然后进入微服务架构的测试部分，探讨微服务的测试挑战和解决方法、监控和扩展常用的实践以及如何将现有架构演变为微服务架构；最后总结微服务架构在设计和开发方面遇到的常见问题及解决方案。

本书既适合想学习微服务架构的开发人员作为入门参考书，也适合有微服务开发经验的技术人员在实践中遇到问题时作为技术手册，查阅最佳实践和解决方案。

译者序

2016 年 11 月，我刚刚加入 ThoughtWorks，就有幸与滕云（《实现领域驱动设计》译者）在同一个项目组工作。滕云与我们分享了领域驱动设计与微服务架构实践之间的潜在联系，例如，如何利用领域驱动设计中的边界上下文（bounded context）对微服务进行划分，如何在微服务设计中识别聚合根（aggregate root），等等。这些分享大大提升了我对微服务架构的兴趣，也纠正了自己对微服务理解不正确的地方。

据了解在 ThoughtWorks“毕业”的标准基本上可以归纳为两个字——出书。彼时我就在想，出书挑战很大，那自己可不可以从翻译一本书入手呢？这样一来可以锻炼自己，为出书提前磨炼一下，二来也至少达到了“肄业”的标准。恰巧在 2017 年 9 月，海侠在群里发了很多需要翻译的书名，让大家自己挑选。其中有一本关于微服务的书让我眼前一亮，于是我立刻回信要求翻译。孰料得到回复说已经有人抢先选走了。正在郁闷之际，好消息传来，我的同事占红来愿意与我共同翻译本书，以飨读者。

本书的前 5 章由 ThoughtWorks 资深开发工程师占红来翻译，而我一直活跃在测试领域，因此负责后 5 章的翻译。之所以这样分工，是因为我们想在各自熟悉的领域准确地理解作者想表达的意图，力求尽可能地贴近原文进行翻译表述。在翻译过程中，尽管我们一丝不苟地进行审查和校对，但仍难免有所疏漏。如果读者在阅读过程中发现某些瑕疵或有争议的地方，非常欢迎联系我们，以便我们进行核对和更正，避免对大家造成困扰和误导。我的联系方式是 hitliubo@gmail.com。

感谢我的同事占红来无私分享了此次翻译的机会，并以翻译界资深人士的身份带我迈进了翻译的大门。感谢他在忙碌的工作之余，指导我共同完成翻译及校对工作。

感谢人民邮电出版社资深高级策划编辑杨海玲女士的大力支持与宽容，她不仅容忍了我们的终极拖延症，还在我们遇到项目安排与翻译时间冲突时给予灵活的调节和疏导。

感谢我的爱人艾华和儿子小午，是你们一直以来的理解和支持才让本书的翻译工作得以完成。

刘博

2018 年 6 月

于西安 ThoughtWorks 办公室

译者简介

占红来是一位咨询师，致力于帮助客户和成就客户。曾主持过某世界 500 强等大型公司的软件一体化开发平台的测试能力提升等落地项目，得到客户的一致好评，并受邀再次合作。

刘博毕业于哈尔滨工业大学，是一位拥有十多年测试经验的软件工程师，主攻自动化测试、性能测试和架构调优领域，对这些领域流行的技术体系和架构风险都有准确的把握。他积极参加对外的技术论坛，把在 IBM、活跃网络和思特沃克的经验积累加以总结并分享给业内相关人士，获得众多好评。

作者简介

Umesh Ram Sharma 是一名软件开发工程师，在可扩展、分布式云服务应用的架构、设计及开发方面有 8 年以上的经验。

他从印度卡纳塔克邦州开放大学获得信息技术专业的硕士学位。出于对微服务和 Spring 的兴趣，他成了 J2EE、JavaScript、Struts、Hibernate 和 Spring 方面的专家，也具有 AWS、J2EE、MySQL、MongoDB、memchached、Apache、Tomcat 和 Hazelcast 等技术的实践经验。

Umesh Ram Sharma 目前是 ZestMoney 公司的首席软件工程师，帮助他的团队将当前项目迁移至微服务。闲暇时，他喜欢开车兜风、烹饪和参加新技术的各种大会。

审阅者简介

Yogendra Sharma 是一名 Java 软件开发工程师，同时拥有 Python 的开发经验，主要精力集中在中间件开发。他拥有计算机科学技术专业的学士学位。

目前 Yogendra 是物联网云架构师，就职于印度浦那的 Intelizign 工程服务（Intelizign Engineering Services）公司。他经常探索技术方面的新鲜事物，并保持开放心态，渴望学习新技术和新框架。

Yogendra 也作为技术审阅人审阅了 *Mastering Python Design Patterns*、*Test-Driven Development with Django*、*Spring 4.0 Microservices*、*Distributed Computing with Java 9*，以及音像课程 *Python Projects*、*Learning Python Data Analysis*、*Django Projects*: *E-Learning Portal*、*Low-Level Python Network Attacks*。以上图书或音像课程均由 Packt 出版社出版。

我想感谢我的父母允许我学习我自己感兴趣的东西，还要感谢我的朋友们的大力支持和鼓励。

前言

在技术世界中，微服务正变得越来越流行，并得到了技术爱好者、开发人员以及许多文章的广泛关注。本书旨在为开发人员提供编写、测试、保护和部署微服务方面的实用指导。使用微服务有很多好处，但也会面临独特的挑战。本书试图以易于理解的方式来阐述使用微服务的最佳实践，同时也阐述了如何避免采用微服务架构可能带来的复杂性陷阱。

各章介绍

第 1 章会介绍微服务架构的整体概念，其中包括对微服务的基本定义。之后会快速浏览本书中所使用的样例程序。

第 2 章将阐明定义微服务组件的指导原则，并说明这些组件如何成为微服务架构的核心支柱。我们会通过 Spring Boot 框架的 Java 项目来有效地定义微服务组件，进而解释这些指导原则如何在实际中使用。最后，通过一个微服务样例来实际演示基于 Java 的微服务组件如何进行配置和服务发现。

第 3 章会讨论并论证微服务间有效通信的原则。之后介绍使用不同技术（如使用 Spring 框架本身或消息代理）实现同步和异步通信的选择标准。我们也会讲到常见问题的最佳处理方法。

第 4 章会讨论使用微服务架构时可能会遇到的常见安全问题和安全挑战。可以通过引入 JWT、OpenID 和 OAuth 2.0 来提高微服务架构的安全性。

第 5 章首先解释基于微服务的数据模型与传统的数据模型的不同之处以及不同的原因。之后会解释数据技术如何被混合使用，以及针对每个微服务组件如何选择适合的数据管理策略。我们还将探索数据模型，解释不同数据模型的选择策略以及选择理由。

第 6 章将探讨测试技术在经常变化和自动部署的系统中的重要性。传统的测试方法和测试标准能完美地匹配微服务架构吗？或者说，我们需要探索完全不同的测试方法来适应微服务架构吗？也许两者都需要。

第 7 章主要着眼于微服务的部署。在微服务架构中，部署操作是非常频繁的，这是

要把部署操作变得尽可能无痛点并且易于使用的一个原因。另外一个原因是，对自动化的追求和对系统扩展性的期望，会导致经常中断并部署新的微服务。Docker 会帮助我们定义微服务的部署流程，并将其自动化。

第 8 章将描述将系统演进至微服务架构的主要机制。这一章还会以示例说明如何演进至微服务架构。

第 9 章会描述监控和扩展基于微服务的系统的重要概念和准则，还会探索监控和扩展基于 Java 的微服务的实际方法，并给出示例说明如何进行监控和扩展。

第 10 章会总结在设计和开发微服务架构时的常见问题，并阐述常见的解决方法或最优方案。

系统需求

推荐使用 Linux 操作系统运行本书的示例，当然也可以使用 Windows 或 Mac 操作系统。但请注意，本书所描述的各种步骤均是以 Linux 为基础的。除此之外，还需要使用 Maven、Java 8 和任意一个 Java 集成开发环境，如 Eclipse、IntelliJ 或 STS。推荐使用 MySQL 社区版作为数据库。

目标读者

本书的目标读者是想开始使用微服务，并希望在实际工作中实践微服务的 Java 工程师。阅读本书的读者无须具备微服务的任何相关知识。

排版约定

在阅读本书时，读者会发现有些文本的样式与其他信息不同。下面是这些样式的样例及其含义。

本书采用以下格式来表示代码、数据库表名、文件夹名、文件名、文件扩展名、路径名、链接、用户输入的信息和 Twitter 用户名："浏览至`/src/main/java/com/sample/firstboot/controller` 文件夹，创建 `SampleController.java` 文件。"

代码清单如下所示：

```
@SpringBootApplication
@EnableZuulProxy
public class ApiGatewayExapleInSpring
{
```

```
    public static void main (String[] args)
    {
        SpringApplication.run(ApiGatewayExampleInSpring.class, args);
    }
}
```

命令行的输入或输出如下所示：

```
curl http://localhost:8888/userService/default
```

新术语或**重要字句**会加粗。例如，我们在使用计算机时见到的菜单或弹出框，在文中的提示格式像这样：“点击 **Switch to the full version**（切换至完整版本）按钮来获取本项目的详细信息。”

警告或重要提示。

技巧和提示。

资源与支持

本书由异步社区出品，社区（https://www.epubit.com/）为您提供相关资源和后续服务。

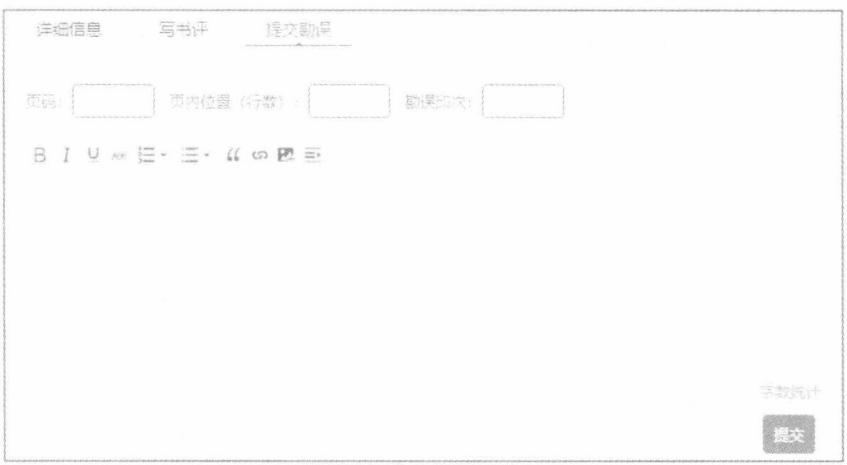
配套资源

本书提供源代码，要获得以上配套资源，请在异步社区本书页面中点击 配套资源，跳转到下载界面，按提示进行操作即可。注意：为保证购书读者的权益，该操作会给出相关提示，要求输入提取码进行验证。

提交勘误

作者和编辑尽最大努力来确保书中内容的准确性，但难免会存在疏漏。欢迎您将发现的问题反馈给我们，帮助我们提升图书的质量。

当您发现错误时，请登录异步社区，按书名搜索，进入本书页面，点击“提交勘误”，输入勘误信息，点击“提交”按钮即可。本书的作者和编辑会对您提交的勘误进行审核，确认并接受后，您将获赠异步社区的 100 积分。积分可用于在异步社区兑换优惠券、样书或奖品。



扫码关注本书

扫描下方二维码，您将会在异步社区微信服务号中看到本书信息及相关的服务提示。

与我们联系

我们的联系邮箱是 contact@epubit.com.cn。

如果您对本书有任何疑问或建议，请您发邮件给我们，并请在邮件标题中注明本书书名，以便我们更高效地做出反馈。

如果您有兴趣出版图书、录制教学视频，或者参与图书翻译、技术审校等工作，可以发邮件给我们；有意出版图书的作者也可以到异步社区在线提交投稿（直接访问 www.epubit.com/selfpublish/submission 即可）。

如果您是学校、培训机构或企业，想批量购买本书或异步社区出版的其他图书，也可以发邮件给我们。

如果您在网上发现有针对异步社区出品图书的各种形式的盗版行为，包括对图书全部或部分内容的非授权传播，请您将怀疑有侵权行为的链接发邮件给我们。您的这一举动是对作者权益的保护，也是我们持续为您提供有价值的内容的动力之源。

关于异步社区和异步图书

“异步社区”是人民邮电出版社旗下 IT 专业图书社区，致力于出版精品 IT 技术图书和相关学习产品，为作译者提供优质出版服务。异步社区创办于 2015 年 8 月，提供大量精品 IT 技术图书和电子书，以及高品质技术文章和视频课程。更多详情请访问异步社区官网 https://www.epubit.com。

“异步图书”是由异步社区编辑团队策划出版的精品 IT 专业图书的品牌，依托于人民邮电出版社近 30 年的计算机图书出版积累和专业编辑团队，相关图书在封面上印有异步图书的 LOGO。异步图书的出版领域包括软件开发、大数据、AI、测试、前端、网络技术等。

异步社区

微信服务号

目录

第 1 章　微服务架构简介……1

1.1　常规微服务架构……2
1.2　微服务架构的特征……2
1.2.1　问题定义……2
1.2.2　解决方案……3
1.3　做好微服务架构面临的挑战……4
1.3.1　通过日志调试……5
1.3.2　服务监控……5
1.3.3　公共库……5
1.3.4　服务之间的消息传递……5
1.3.5　微服务的部署和版本管理……6
1.4　微服务的未来……6
1.4.1　无服务架构……7
1.4.2　微服务即 PaaS……7
1.5　与传统架构相比微服务架构的优势……7
1.6　是不是看起来与 SOA 很像……9
1.7　将业务领域划分为微服务组件……11
1.8　到底要不要使用微服务……13
1.8.1　组织认同度……13
1.8.2　体验 DevOps……14
1.8.3　分析现有数据库模型……14
1.8.4　自动化和 CI/CD……14
1.8.5　集成……14
1.8.6　安全……14
1.8.7　成功迁移的例子……15
1.9　示例项目（信用风险评估引擎）……15
1.10　Spring……15
1.11　Spring Boot……16
1.12　小结……19

第 2 章　定义微服务组件……21

2.1　微服务的定义……21
2.2　服务发现及其用途……22
2.2.1　DNS……22
2.2.2　服务发现的请求……23
2.2.3　服务发现模式示例……26
2.2.4　整个架构中的配置外化……29
2.3　API 网关及其诉求……32
2.3.1　认证鉴权……33
2.3.2　不同协议……34
2.3.3　负载均衡……34
2.3.4　请求转发（包括服务发现）……34
2.3.5　响应转换……34
2.3.6　断路器……35
2.3.7　API 网关的优劣性……35
2.4　API 网关的例子……36
2.5　用户注册微服务的开发……39
2.6　小结……59

第 3 章　微服务端点之间的通信……61

3.1　微服务间应该如何通信……61
3.2　编制和编排……62
3.2.1　编制……62
3.2.2　编排……63
3.3　同步通信和异步通信……65
3.3.1　同步通信……65
3.3.2　异步通信……68
3.3.3　财务服务……79
3.4　小结……90

第 4 章　微服务端点的安全……91

4.1　微服务的安全挑战……91
4.1.1　复合技术栈或者存在遗留代码……92
4.1.2　认证和授权（访问控制）……92
4.1.3　基于令牌的安全实践……92
4.1.4　安全性的责任……92
4.1.5　编制风格的风险……92
4.1.6　微服务之间的通信……93
4.2　与 OpenID 的 OAuth 2.0 一起使用 JWT……94
4.2.1　OpenID……94
4.2.2　OAuth 2.0……95
4.2.3　JWT……97
4.2.4　示例应用……99
4.3　小结……111

第 5 章　创建高效的数据模型……113

5.1　数据和建模……113
5.2　单体架构中的数据模型……114
5.3　SOA 中的数据模型……115
5.4　微服务架构中的数据模型……116
5.4.1　每个微服务限定一些数据库表……116
5.4.2　每个微服务一个数据库……117
5.4.3　Saga 模式……117
5.4.4　必要时采用混合数据技术……119
5.5　从单体应用向微服务迁移数据模型……120
5.5.1　领域驱动设计……120
5.5.2　数据模型迁移方式……121
5.6　小结……126

第 6 章　测试微服务……127

6.1　微服务中测试的目的……127
6.2　单元测试……128
6.3　集成测试……131
6.4　组件（服务）测试……131
6.5　契约测试……133
6.5.1　Pact……133
6.5.2　Spring Cloud Contract……134
6.6　端到端测试……135
6.7　更进一步……135
6.8　小结……136

第 7 章　部署微服务……137

7.1　持续集成……137
7.2　持续交付……138
7.3　用微服务配置 CI 和 CD 工具……140
7.4　微服务的 Docker 化……147
7.4.1　Docker……148
7.4.2　Docker 引擎……148

7.4.3 Docker 镜像 ····· 148
7.4.4 Docker 存储 ····· 148
7.4.5 应用程序在 Docker 中是如何工作的 ····· 149
7.4.6 公共、私有和官方的镜像库 ····· 149
7.4.7 Docker 与 VM 的区别 ····· 149
7.4.8 在 Linux 中安装 Docker ····· 150
7.5 在 Docker 化的微服务中使用开源 CI 工具 ····· 154
7.6 小结 ····· 156

第 8 章 演进现有系统 ····· 157

8.1 从哪里开始 ····· 159
8.1.1 架构视角和最佳实践 ····· 159
8.1.2 数据库视角和最佳实践 ····· 162
8.2 示例应用及其演变过程 ····· 163
8.2.1 用户管理服务 ····· 164
8.2.2 购物车/订单服务 ····· 164
8.2.3 支付服务 ····· 164
8.2.4 配送/跟踪服务和通信服务 ····· 164
8.2.5 产品推荐服务 ····· 165
8.2.6 调度服务 ····· 165
8.3 小结 ····· 166

第 9 章 微服务的监控和扩展 ····· 167

9.1 微服务系统的监控原则 ····· 167
9.1.1 如何设置并使用警报 ····· 168
9.1.2 从一开始做好监控和发布渠道规划 ····· 168
9.1.3 自动扩展和自动发现 ····· 168
9.1.4 监控前门流量 ····· 169
9.2 监控模式的发展变化 ····· 169
9.3 日志记录有助于监控 ····· 170
9.4 微服务系统的扩展原则 ····· 171
9.4.1 *x* 轴 ····· 172
9.4.2 *y* 轴 ····· 172
9.4.3 *z* 轴 ····· 173
9.5 实施扩展策略前请三思 ····· 174
9.6 微服务的监控和扩展工具 ····· 175
9.7 小结 ····· 180

第 10 章 故障排除 ····· 181

10.1 使用微服务时的常见问题 ····· 181
10.1.1 性能下降 ····· 181
10.1.2 日志记录位置因编程语言而异 ····· 183
10.1.3 多组件之间的耦合或依赖问题 ····· 184
10.1.4 服务部署数量与日俱增 ····· 184
10.1.5 监控多项服务，发现性能下降或其他问题 ····· 185
10.1.6 日志与不同组件的关系 ····· 185
10.2 常见问题的解决方法 ····· 186
10.2.1 解决性能问题的步骤 ····· 186
10.2.2 处理不同语言生成的并处于不同位置的日志记录 ····· 186
10.2.3 服务之间的依赖关系 ····· 187
10.2.4 DevOps 专家积极参与 ····· 187
10.2.5 监控 ····· 188
10.3 小结 ····· 189

第 1 章
微服务架构简介

软件架构可以定义为系统设计的一组规则和原则，它定义了软件系统的元素、行为、结构和不同组件之间的关系。

在 20 世纪 80 年代初期，出现了一些大型软件系统，亟需一种统一的模式（也就是后来的架构）来解决设计这些庞大系统所面临的一些常见问题。从那时开始，演化出了今天我们所熟知的“软件架构”的概念。自此之后，很多架构类型被引入到大型软件系统的设计当中。细细数来，软件行业已经见证了**从不共享架构**（shared nothing），到**单体架构**（monolithic），到**客户-服务器架构**（client-server），到**分布式多层架构**（n-tire），再到**面向服务架构**（service-oriented architecture，SOA）等架构风格。**微服务架构**无疑就是这条演化链上的一个新节点。

近年来，微服务这个词的热度在各种软件开发者/架构师社区中呈指数级增长。我们经常听到一些采用了单体架构的组织抱怨发布周期太长、调试烦琐、维护成本高、扩容难等问题。这些问题罄竹难书，以至于即使是少数管理得很好的单体应用也需要花费大量的人力物力来解决这些问题。微服务为解决这些问题提供了一种高效的办法，这也毫无疑问是其日益火热的原因之一。一言以蔽之，微服务架构可以把一个很大、很复杂的问题分解成一系列相对较小的服务，并且每个服务只负责自己分管的那一部分。

微服务架构的基本哲理是：*只做一件事，并把它做到极致。*

微服务的核心是**单一职责原则**（Single Responsibility Principle，SRP）。在微服务架构中，大的业务块会被拆分为一些小的任务，每一个小的任务都依托于一个微服务来完成。在微服务架构的系统中，微服务的数量可多可少，取决于业务需求以及任务被拆分的情况。微服务架构可以给组织带来很多单体架构所没有的好处，但是同时，微服务架构也有自己的一些问题需要解决。我们会在接下来的章节中继续讨论微服务架构的优势和短板。