Broadview
www.broadview.com.cn

[PACKT] open source
PUBLISHING
community experience distilled

Developing Micro[redacted]de.js

Node.js微服务

本书是以Node.js作为服务端编程语言来讲解微服务开发的实践指南，
通过阅读本书可以帮助你开发出高效且可伸缩的微服务

[美] David Gonzalez 著
赵震一 郑伟杰 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

Developing Microservices with Node.js

Node.js微服务

[美] David Gonzalez 著
赵震一 郑伟杰 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书对如何采用 Node.js 及其生态工具进行微服务开发的最佳实践做了全面的介绍，内容包括对微服务架构基本概念及设计原则的讲解，以及如何采用 Node.js 搭配 Seneca、PM2 和 Docker 等现代化工具来构建、测试、监控以及部署轻量级微服务，同时也阐述了 Node.js 在微服务实践中所涉及的相关概念，并就微服务的优缺点、文档化、安全性以及可追溯性等主题进行了探讨。

本书适合掌握服务端开发基本知识的 Node.js 开发者以及使用 Java、C#等其他服务端技术栈并对微服务实践感兴趣的所有开发者。

版权贸易合同登记号　图字：01-2016-7164

图书在版编目（CIP）数据

Node.js 微服务/（美）大卫·冈萨雷斯（David Gonzalez）著；赵震一，郑伟杰译. —北京：电子工业出版社，2017.1

书名原文：Developing Microservices with Node.js

ISBN 978-7-121-30524-5

Ⅰ. ①N… Ⅱ. ①大… ②赵… ③郑… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2016）第 290015 号

策划编辑：张春雨
责任编辑：刘　舫
印　　刷：三河市良远印务有限公司
装　　订：三河市良远印务有限公司
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：787×980　1/16　　印张：16　　字数：320 千字
版　　次：2017 年 1 月第 1 版
印　　次：2017 年 1 月第 1 次印刷
定　　价：69.00 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：010-51260888-819　faq@phei.com.cn。

译者序

在技术圈里，微服务已不是一张生面孔，确切地说，如今已算得上是一名当红的明星了。然而，对于任何一门流行的技术而言，从出现到被广泛采纳必定都会经历一个被质疑、被挑战，以及在实践的锤炼中不断进化的过程，微服务也是如此。

作为本书的译者，我并不打算在这篇序里给出太多的剧透。但是对扒一扒“微服务”的成长史却深感义不容辞。

根据 Martin Fowler 大叔的回忆，“Microservices”一词是在 2011 年 5 月于威尼斯附近举办的一次架构师工作坊的讨论中被首次提出的。

2012 年，微服务正式出道。“Microservices”一词首次在 ThoughtWorks 技术雷达 2012 年 3 月的报告中亮相。当时报告对其成熟度的评级位于“评估（Assess）”象限。

不到一年时间，“Microservices”一词在 2012 年 10 月的技术雷达中已经进入了“试验（Trial）”象限。这份报告称，TW 及更广范围内的社区都将微服务作为一项分布式系统设计的技术开始采用。

2013 年，可配合微服务实施的一些框架和工具相继出现，比如 Spring Boot、Hystrix 等。

在此之后，业界对于微服务的实践及讨论逐渐升温。2014 年 3 月，Adrian Cockcroft（前 Netflix 首席云架构师，被誉为“让 Netflix 走向云端的男人”）与 John Allspaw（现任 Etsy 的 CTO）等人在 Twitter 上展开了关于“微服务与单块应用”孰优孰劣的讨论[1]。

而在业界，真正为“微服务架构”这一架构风格正名的当属 Martin Fowler 大叔于 2014

[1] https://www.infoq.com/news/2014/08/microservices-monoliths

年 3 月在其博客发表的 *Microservices*[2]一文，也正是此文让大众对微服务有了更加具体的认识。

2015 年，随着以 Docker 为代表的容器技术的突飞猛进，微服务的部署难题迎刃而解，甚至有人将 2015 年称为微服务架构元年。

而当我们跨入 2016 年甚至是 2017 年的时候，微服务已正值壮年。在书店及互联网上，关于什么是微服务的书籍、博文已成燎原之势。对于那些希望了解微服务“是什么”的人来说，这是一个美好的时代。但是就微服务生命周期各个阶段该“怎么做”而言，译者深感始终缺少一本接地气的实践指南。

当电子工业出版社计算机出版分社的张春雨编辑向我推荐这本书的时候，我的心里是纠结的。因为我目前本职的工作量是相当饱和的（老板请看过来~），但是却无法拒绝这本书。这不就是那部我寻找已久的接地气的“missing guide”吗？

尤其是将“微服务”与“Node.js”这两味如此珍贵的药材一起入药时，它们又会对“单块系统”中的哪些痼疾产生怎样奇特的疗效呢？说好了不剧透，那么就请读者亲自体会这一段接地气的技术之旅吧。

再次感谢张春雨编辑对我的信任，也非常荣幸能参与这本书的翻译。当然，翻译一本书并不是一件轻松的事情，我要感谢我的师弟郑伟杰，他与我共同承担了本书的翻译工作，正因为有了他的加入才让我得以工作翻译两不误。其次要感谢伟杰的女友，她为本书译文做了审阅与润色。最后，我要感谢我的家人，尤其是我的老婆和父母，你们是我坚强的后盾，让我能专注于做好自己喜欢的事情。

由于时间及能力所限，我们对于原书的理解及对译文的表述难免存在一些不妥之处，希望各位读者给予理解及反馈。我的邮箱是 emsn1026@gmail.com，欢迎各位读者与我们联系。

赵震一

2016 年 8 月于杭州

[2] http://martinfowler.com/articles/microservices.html

关于作者

David Gonzalez 是一名在编程语言方面“极不专一”的软件工程师，他在金融服务行业“混迹”多年。他尝试找到抽象层次合适的解决方案，并探索着如何保证既不过于具体也不过于抽象之间的平衡。

David 曾求学于西班牙，但是不久之后便转战都柏林，自 2011 年起便定居于此并开启了更为宽广和有趣的职业生涯。他目前是一名金融技术领域的独立咨询师。他的 Linkedin 账号地址是：`https://ie.linkedin.com/in/david-gonzalez-737b7383`。

David 乐于尝试新的技术和范式，从而能让自己在软件开发的复杂世界中不断拓展出新的版图。

献给我的妻子 Ester，感谢你在生活中的方方面面都给予我无条件的支持。

献给我尚未出生的女儿 Elena，愿生活带给你所有的快乐，一如你会将这些快乐同样带给我们全家。

关于审校者

Kishore Kumar Yekkanti 是一名经验丰富的专家，他在过去的十年里曾与不同的领域和技术打过交道。他对软件开发中的消除浪费尤具热情。Kishore 是敏捷原则的巨大贡献者和遵循者。他是一名善于开发端到端系统的全栈开发者，同时也是一名通晓多种语言的程序员。目前他专注于高度分布式应用中的微服务扩展，而这些应用部署于云端基于容器的系统(Docker)之中。他曾在多家知名的公司担任过首席工程师，这些公司包括 Thoughtworks、CurrencyFair 等。他曾通过微服务为这些公司的团队带来新生。

献给我的搭档和挚友 Jyothsna，以及我的女儿 Dhruti，尽管我的工作日程安排得是那么疯狂，她却一如既往地迁就着我。

前言

作为一本微服务入门的实践指南，本书采用了 Node.js 和以 Seneca、PM2 为主的现代框架来进行阐述。在各章中，我们将分别介绍如何利用最佳实践去设计、构建、测试以及部署微服务。此外，我们还会讨论另外一个有价值的课题——如何在设计系统时做出合理的妥协，来避免过度设计和确保技术方案与实际业务需求相匹配。

本书概述

第 1 章主要讲述微服务的基本概念，包括主要优点和一些缺点，本章内容是本书后续章节的基础。

第 2 章介绍了 Node.js、Seneca 和 PM2。还讨论了 Node.js 应用的结构，以及如何通过 PM2 来运行应用。另外，我们还研究了一些 Seneca 与 PM2 的替代产品。

第 3 章主要讲述如何使用微服务来处理自然增长（计划之外的软件需求变更）。另外，我们还讨论了如何将单块应用分解成微服务。

第 4 章阐述了如何编写我们的第一个微服务程序。

第 5 章涉及了安全性与可追溯性，这是现代系统的两大重要特性，因为我们需要保证信息的安全与操作的可追溯性。在本章中，我们讨论了使用 Seneca 来保证安全性与可追溯性的方法。

第 6 章主要介绍了 Node.js 的两大主流测试框架——Mocha 和 Chai。同时使用 Sinon 来 mock 服务以及 Swagger 来为微服务进行文档化。

第 7 章使用 PM2 结合 Keymetrics 来监控微服务，使 PM2 的功能得到最大发挥。

第 8 章通过使用 PM2，学习如何在不同环境下部署微服务，并通过单条命令管理应用的“生态系统”，从而减少微服务架构带来的开销。我们还将讨论 Docker，它是一个应用容器引擎，可以部署包括 Node 应用在内的各种应用。

阅读本书的准备工作

为了能够完成本书的实践案例，需要预先安装 Node.js、PM2（可以通过 npm 来安装），以及 MongoDB。此外还需要一个编辑器，我个人选用了 Atom，但是一般通用的编辑器都能满足需求。

本书的读者对象

本书适合具有一定 Node.js 经验，并且想要学习 Seneca 以及微服务知识的开发者。在本书中，有 70%的内容是面向实践的（因此我们会编写大量代码），有 30%是理论知识。基于编写的这些代码可以帮助读者将书中提到的模式应用到新的软件开发中去。

约定惯例

本书将会使用不同的书写风格来区分不同种类的信息。以下是这些风格的例子和它们的意义。

正文中的文本代码、数据库表名、文件夹名、文件名、文件扩展名、路径名、URL、用户输入和推特用户定位（Twitter handles）将会用代码体书写，如“我们知道输入参数是一个 `PaymentRequest` 实例”。

代码块则将会是这样的风格：

```
public interface PaymentService {
    PaymentResponse processPayment(PaymentRequest request) throws
        MyBusinessException;
}
```

如果希望向你强调代码块中的一部分，那么它们将会以粗体展示：

```
function() {
```

```
    sinon.stub(Math, "random");
    rollDice();
    console.log(Math.random.calledWith());
});
after(function(){
    Math.random.restore();
});
```

任何命令行的输入和输出将是以下这样的：

```
node index.js
npm start
```

警告和关键提醒将会在这样的图标后出现。

小提示和小技巧将会在这样的图标后出现。

下载示例代码

你可以从 http://www.broadview.com.cn 下载所有已购买的博文视点书籍的示例代码文件。

勘误表

虽然我们已经尽力谨慎地确保内容的准确性，但错误仍然存在。如果你发现了书中的错误，包括正文和代码中的错误，请告诉我们，我们会非常感激你。这样，你不仅帮助了其他读者，也帮助我们改进后续的出版。如发现任何勘误，可以在博文视点网站相应图书的页面提交勘误信息。一旦你找到的错误被证实，你提交的信息就会被接受，我们的网站也会发布这些勘误信息。你可以随时浏览图书页面，查看已发布的勘误信息。

目录

1 微服务架构

微服务日趋流行。时下，几乎每一个参与“绿地项目”[译注1]的工程师都应该考虑采用微服务来提升所建系统的质量，他们应该熟悉涉及这类系统的架构原则。我们将会在本书中揭示微服务与面向服务架构（SOA）之间的区别。同时还会向大家介绍一款用以编写微服务的强大平台——Node.js。通过采用 Node.js，我们可以轻易地创建出高性能的微服务。

本章将会带领你从架构的视角来审视微服务：

- 什么是微服务
- 面向微服务的架构
- 关键的好处
- SOA 与微服务的比较
- 为什么选择 Node.js

微服务应运而生

在过去 40 年里，软件开发的世界日新月异。在这一飞速发展的过程中，一个关键的增长点便是系统的规模。从古老的 MS-DOS 发展到当代的系统，规模有了数百倍的增长。这种飞跃式的增长迫使我们不得不去寻求更好的方式来组织代码及软件组件。通常，在一家公司随着业务需求的增长而逐步发展（即自然增长）的过程中，前期往往是以单块架构（monolithic architecture）的方式来组织系统的。因为对于软件的初期构建来说，单块架构

译注1 绿地项目（green field）意指那些没有太多前置约束的项目。类比自在一片绿地上开发项目，没有既有建筑与基础设施的约束。

的方式是最容易且最高效的。但是若干年（甚至是几个月）后，受限于前期既有单块软件系统内部的耦合性，向该系统添加新功能变得越来越艰难。

单块软件

如今，对于像 Amazon 和 Netflix 这样的高新技术公司来说，采用微服务来构建新的软件系统已经是大势所趋。理想情况下，面向微服务的软件在帮助这些公司扩展新产品规模方面有着强大的优势（学习完本书后，你也将深谙此道）。现在的问题是，并不是所有的公司都能提前对软件系统做好规划。相比于提前规划，很多公司一直以来都是基于自然增长的方式来构建软件系统的，鲜有软件组件会根据业务关系的紧密度来对业务流进行分类。不难发现，大多数公司都只有两个大的软件组件：面向用户的网站和内部的管理系统。我们通常将这种架构方式称为单块软件架构。

一部分这类公司在尝试扩充工程团队的时候，面临了巨大的挑战。协调好多个团队来对单个软件组件进行构建、部署以及维护是一件相当艰巨的任务。系统发布与 bug 重复引入之间的冲突往往是家常便饭，这些问题耗费了团队大量的精力。解决该问题的一个方案（同时还会带来额外的好处）是将单块软件拆分成微服务。每个团队都专职从事某个相对较小模块的维护，且这些组件也都是自治且互相隔离的。这样一来，我们便可以独立地对这些组件进行版本化、更新以及部署，从而不会牵扯公司的其他系统。

将庞大的单块系统拆分成微服务可以让工程团队创造出互相隔离且独立自治的工作单元。这些工作单元从功能上来说都是高度专职化的，例如可以专职处理电子邮件的发送、支付卡交易等特定任务。

现实世界中的微服务

微服务是指那些小型的软件组件，每个服务都专职处理某一类任务，将它们有机整合后便可用于处理更高级的复杂任务。让我们先忘掉软件这个概念，然后回过头来思考一下一家公司是如何运作的。当某个候选人申请公司的某份工作时，他应聘的是一个特定的职位，比如：软件工程师、系统管理员或者办公室经理。之所以这样的原因归根结底就是一个词：专职化（specialization）。如果你曾任职软件工程师，将会在软件开发方面具有更丰富的经验，并且能在这方面给公司带来更多的价值。事实上，你并不知道应该如何跟客户打好交道，但是这并不会影响你的绩效，因为这并不是你的专业领域，所以在这方面你很难在日常工作中产出更多的价值。