详解Spring Cloud所提供的微服务架构核心组件

详解微服务架构从设计、开发、部署到运维的一站式解决方案

通过大量易于构建、运行和测试的开发示例，带领读者实战微服务架构

Spring Cloud
微服务架构开发实战

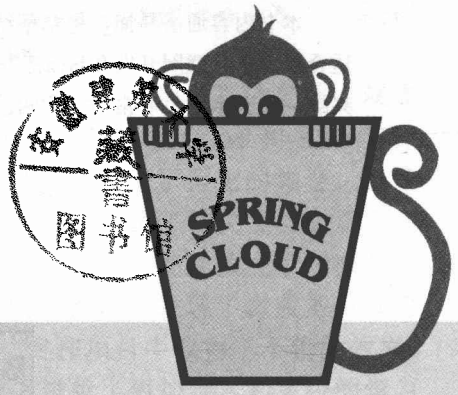
董超　胡炽维◎编著

- 详解Spring Cloud核心组件：服务发现、客户端负载均衡、API网关、微服务容错、统一配置中心、消息总线及微服务调用监控等
- 手把手带领读者使用Spring Boot进行微服务应用开发
- 手把手带领读者使用Config组件实现统一配置管理及加密处理
- 手把手带领读者使用Kafka和Redis构建基于消息驱动的应用
- 手把手带领读者使用OAuth 2.0和JWT构建安全解决方案
- 手把手带领读者使用Docker和Jenkins实现微服务应用的自动化部署

机械工业出版社
China Machine Press

Spring Cloud
微服务架构开发实战

董超　胡炽维◎编著



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Spring Cloud微服务架构开发实战/董超，胡炽维编著. —北京：机械工业出版社，2018.7

ISBN 978-7-111-60452-5

Ⅰ. S… Ⅱ. ①董… ②胡… 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（2018）第156132号

本书首先从微服务架构兴起的背景讲起，探讨了为何在分布式系统开发中微服务架构将逐渐取代单体架构，然后对Spring Cloud所提供的微服务组件及解决方案进行了一一讲解，从而让读者不但可以系统地学习Spring Cloud的相关知识，而且还可以全面掌握微服务架构应用的设计、开发、部署和运维等知识。

本书共11章，分为3篇。第1篇为微服务开发基础——Spring Boot框架及使用；第2篇为Spring Cloud组件实战；第3篇为微服务与Docker容器技术。其中第2篇为全书的核心，涵盖了构建微服务架构所需要的服务治理（Eureka）、客户端负载均衡（Ribbon）、微服务容错与降级处理（Hystrix）、微服务API统一网关（Zuul）、分布式配置中心（Config）、微服务调用链追踪（Sleuth）、微服务消息驱动开发（Stream）及微服务安全（OAuth及JWT）等相关知识。

本书内容通俗易懂，每章都结合实例进行讲解，特别适合Spring Cloud的入门读者阅读，也适合致力于互联网开发和Java开发的进阶读者阅读。如果你是运维人员，或者你对微服务架构有兴趣，那么本书也非常适合你阅读。此外，本书也可以作为相关培训机构的教材使用。

Spring Cloud 微服务架构与开发实战

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：欧振旭 李华君 责任校对：姚志娟
印 刷：中国电影出版社印刷厂 版 次：2018 年 8 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：20.75
书 号：ISBN 978-7-111-60452-5 定 价：79.00 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

前言

2014 年 3 月，Martin Fowler 在其博客上发表了 Microservices（微服务）一文，对过去几年逐渐开始流行的微服务架构开发模式给出了正式的定义。同年，Netflix OSS 将自己多年来在实际开发中所使用的微服务基础组件开源。随后，Pivotal 在 Netflix OSS 的基础上对这些组件进行了封装和集成，推出了 Spring Cloud。到如今，微服务架构已逐渐普及，在技术生态上也得到了不断完善和更新，其在容器、应用框架、发布管理及监控等方面都有了长足进步。微服务在日常开发中也崭露头角，逐渐得到了开发者的认同。与此同时，Spring Cloud 在过去几年中快速发展，不断迭代和更新，已经形成了微服务开发“全家桶”式的解决方案，逐渐在微服务开发领域奠定了坚实的基础。

当我第一次接触微服务的概念时，以为这又是一个“新瓶装旧酒”的把戏。就像当年一些大公司为解决分布式大型应用所提出的 COBRA、EJB、DCOM 和 ESB 等概念一样难以使用，它们甚至还没有来得及等用户完全掌握就退出了历史的舞台。当我再次注意到微服务时，是因为 Spring Cloud 的推出。在大致阅读了相关文档之后顺便“跑”了一下示例项目，我就被其深深吸引了。正是这次接触，使我改变了对微服务的看法。正巧接下来的一段时间，公司内部正在做架构调整，也发现了微服务有其可用之处，所以便在架构调整中引入了 Spring Cloud，并最终在项目中取得了不错的效果。这加深了我对微服务的好感，所以后续对 Spring Cloud 进行了更加深入和系统的学习与研究。

不过话又说回来，虽然公司的第一次微服务架构实践取得了不错的效果，但在实践中也出现了很多问题。感受最深的是，微服务架构不再仅仅是编写代码。单体架构应用开发，一般在完成代码编写之后部署上线就可以了。但在微服务架构下，项目的部署和运维等都面临着新的考验。幸好，Spring Cloud 本身提供了一系列开箱即用的功能，可以帮助开发人员降低微服务实践的难度。

这两年，图书市场上虽然已经有了一些与微服务相关的图书，但却鲜见一本比较系统、实用，能够真正解决开发人员实际问题的微服务架构图书。基于此原因，我下定决心写一本书，通过讲解 Spring Cloud 为微服务开发所提供的核心组件，带领读者进入微服务开发的世界，并将 Spring Cloud 中涉及的微服务核心解决方案及微服务模式通过示例的方式一一呈现给读者，以期解决开发人员的实际问题。

本书特色

- 本书内容丰富，不仅涵盖了 Spring Cloud 的核心组件，而且还介绍了如何通过 Spring Boot 来搭建微服务，并介绍了 Kafka、ELK 和 Redis 等流行技术。
- 书中对微服务架构中的“痛点”，比如安全和消息应用等，都有非常详细的讲解，可以让读者快速掌握如何通过 OAuth 2.0 和 JWT 构建单点登录安全解决方案。另外，本书还详细介绍了如何使用 Kafka 和 Redis 构建基于消息驱动的应用。
- 本书在讲解时给出了大量的开发示例，这些示例通俗易懂，且易于构建、运行和测试，能够让读者在学习微服务架构时快速进入实战，从而对 Spring Cloud 所提供的组件有直观认知。
- 本书通过实例来讲解微服务自动化部署解决方案，可以让读者不仅能够掌握微服务开发的知识，还可以学习微服务部署和运维的知识，从而全面理解微服务架构之道。

本书内容

本书共 11 章，分为 3 篇。书中首先从开发微服务所使用的基础框架 Spring Boot 开始讲起，然后重点讲述了 Spring Cloud 中的核心组件，最后介绍了微服务部署的相关技术。

第1篇　微服务开发基础——Spring Boot框架及使用（第1、2章）

第 1 章介绍了微服务架构开发的基础概念，并对比传统单体架构开发，总结了微服务架构的优缺点，以及如何将传统的应用拆分和迁移到微服务架构体系下，并列出了微服务架构的开发原则。

第 2 章讲述了微服务架构开发的基础——Spring Boot，并通过 Spring Boot 技术构建了贯穿本书的一个项目案例——电子商城的单体应用案例。通过对该项目案例的构建，讲解了开发时需要的基础知识，如 RestController、JPA 和 API 文档自动生成等技术。此外，本章还讲解了 Spring Boot 的一些特性，如自动配置机制和扩展机制等，这些特性在应用开发中都非常有用。

第2篇　Spring Cloud组件实战（第3~10章）

第 3 章探讨了微服务架构开发需要关心的关键特性及业界的解决方案，并给出了 Spring Cloud 子项目。本章的最后一节给出了 Hello World 经典示例程序，读者可以感受一下通过 Spring Cloud 构建微服务的威力。

第 4 章介绍了微服务架构最重要的一个服务支持组件：服务治理（Eureka）。通过介绍 Eureka 服务治理，可以让读者了解服务提供者如何将微服务注册到服务治理服务器中，以及服务消费者如何通过服务治理服务器来调用服务。在服务治理技术的基础上，本章还

讲解了客户端负载均衡的实现方式及如何通过 Feign 实现声明式服务调用。

第 5 章讲解了当微服务远程调用失败时如何启动服务应急预案——服务降级处理（Hystrix）。通过服务降级处理可以避免因调用失败而引起的“雪崩效应”。本章首先讲解了如何使用 Spring Cloud Hystrix 实现服务降级处理；然后对 Hystrix 执行流程及相应的“断路器”模式进行了详细分析，并给出了实施服务降级处理时可以参考的模式；最后引入了可视化分析监控，对微服务中各个服务及相应的方法进行统计分析。

第 6 章讲述了如何通过 Spring Cloud 所提供的 Zuul 组件来实现 API 服务网关。通过 API 服务网关可以对后端具体实现细节进行隐藏，并且能够简化前端的调用。本章详细讲述了 Zuul 所提供的路由映射和请求过滤两个功能，并对路由映射规则进行了详细讲解。对于 Zuul 的请求过滤功能，重点讲述了过滤流程和几种标准的过滤器用法。通过这些过滤器，可以在 API 服务网关中统一实现用户身份验证、业务权限鉴权、流量与并发控制等处理，从而避免在每个微服务中重复实现这些处理。

第 7 介绍了如何将前面章节中的项目案例的配置统一迁移到配置资源库中进行管理，并通过 Spring Cloud Config 实现配置文件的加载和管理，从而实现在项目中不需要重复编写配置文件。本章还结合 Git 讲述了如何让配置文件具有版本管理的功能，以及如何对项目中的敏感配置信息进行加密和解密，并施行配置文件的动态刷新。

第 8 章介绍了微服务架构中比较让人头疼的一个问题：服务链路跟踪解决方案（Sleuth）。通过在项目中嵌入 Sleuth，可以一次性将用户服务请求所调用的所有微服务串联起来，从而掌握微服务之间的关系。本章剖析了 Sleuth 的基础原理，并通过 Zipkin 服务器对服务调用链路进行了可视化分析，然后在此基础上讲述了如何与 ELK 整合，从而将微服务日志集中在一起，避免了当排查问题时需要跨越多个服务器进行查看的弊端。

第 9 章介绍了如何在微服务之间实施面向消息驱动的开发（MoM），并基于一个翔实的例子讲述了消息的构建、发送和消费的整个流程。本章通过 Spring Cloud 的 Stream 项目实现了与 Kafka 消息中间件的对接，并讲解了如何使用“发布-订阅”模式。本章最后通过 Spring Cloud 基于 Stream 之上所提供的另外一个项目——Bus，实现了 Config 的自动刷新处理。

第 10 章讲述了微服务安全的相关内容。本章首先介绍了 Spring 提供的 Spring Security 框架；然后根据微服务架构的特点，介绍了如何基于 Spring Cloud Security 来实现 OAuth 2.0 协议的用户单点登录功能；最后再进一步将安全服务器所生成的用户访问令牌转换成 JWT 格式的令牌，从而让每一个微服务可以避免多次与认证服务器之间进行交互，提升了微服务架构应用的弹性和效率。

第3篇　微服务与Docker容器技术（第11章）

第 11 章首先介绍了微服务部署所需要的基础知识——Docker，读者掌握了有关 Docker 的知识后，便能够将编写的微服务部署到 Docker 容器中；然后介绍了如何使用自动构建部署工具——Jenkins，通过配置 Jenkins 的构建任务，可以实现对微服务源码变更的监控并进行自动发布，从而将开发者从重复、乏味的编译、打包部署中解放出来；最后简要介绍了

微服务的集群管理和编排。

本书源代码获取方式

本书涉及的所有源代码文件都可以从 https://github.com/cd826/springcloud-demo 网站上下载。这些源代码按照书中的章节进行组织，并在源代码的 README.md 文件中给出了说明。如果某个章节中有多个不同的示例需要拆分成不同的项目，为了方便读者使用，将其分成了多个文件夹，而且在每一个文件夹中都包含了该示例所需要的全部工程代码。这些代码都可以使用 Maven 组织管理，并且可以在 IDE 之间使用根文件夹中的 pom.xml 导入整个工程，从而对示例代码进行编译、打包和测试，而不需要一个个地导入。

另外，书中涉及的所有源代码文件还可以在机械工业出版社华章公司的网站 www.hzbook.com 上获取。请在该网站上搜索到本书，然后找到资料下载模块即可下载。

书中涉及的所有示例代码均在 JDK 1.8 和 Maven 3.3.9 版本下测试通过。如果读者在编译时产生错误，需要仔细检查一下部分注释掉的配置和代码。因为这些被注释掉的配置和代码可能是为了配合书中演示的某些功能而被注释掉的。

此外，书中用到的工具和相关服务器环境等都可以在其官网上下载，读者可自行下载，本书源码中并不包含这些内容。

本书适合哪些读者阅读

- 如果你是一名 Java 程序员，并且打算构建分布式系统，那么本书适合你阅读；
- 如果你对微服务开发非常有兴趣，那么本书适合你阅读；
- 如果你对使用 Spring Cloud 开发云原生应用非常感兴趣，那么本书适合你阅读；
- 如果你正打算开发一款互联网应用产品，那么本书适合你阅读；
- 如果你想知道 Spring Cloud 中各子项目在微服务中的角色及位置，那么本书适合你阅读；
- 如果你想了解基于微服务架构的开发对运维和部署的影响，那么本书适合你阅读；
- 如果你是 Spring 技术的狂热分子，那么本书非常适合你阅读。

售后服务

读者阅读本书时若有疑问，或者有好的建议可以反馈给笔者或编辑，我们会抽空回复。

由于作者的水平所限，加之时间有限，书中可能还存在一些疏漏和不当之处，也敬请各位读者斧正。

联系我们请发电子邮件到 hzbook2017@163.com。

董超

目录

第 2 篇 Spring Cloud 组件实战

第 3 篇　微服务与 Docker 容器技术

第 1 篇
微服务开发基础——Spring Boot 框架及使用

第 1 章　微服务架构开发

从软件开发诞生之初，业界一直致力于寻找大型分布式应用系统开发的“银弹”，从结构化编程、面向对象，到 COBRA、EJB、ESB、SOA 等。Fred Brooks 在《没有银弹：软件工程的本质性与附属性工作》一文中提出：大型分布式系统具有复杂性、隐匿性、配合性和易变性四大难题，不会存在任何单一软件工程上的突破，能够让开发生产力得到一个数量级上的提升。而微服务的出现虽然不是一颗“银弹”，但是它给出了四大难题的对策，解决了以往单体架构系统构建的困境。

1.1　单体架构应用的困境

在 Java 开发中，一个典型的单体架构应用就是将一个应用中所有的功能都打包在一个 WAR 文件中，并部署在应用服务器（如 Tomcat）中运行，如图 1-1 所示。

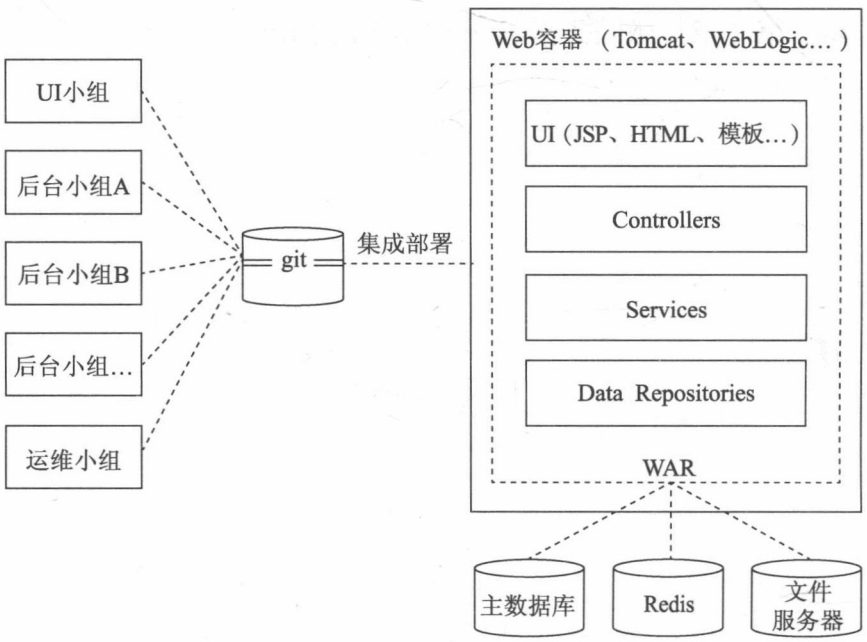


图 1-1　典型单体架构应用结构图

对于单体架构应用来说，随着业务的扩张，其开发、部署和运维都会越来越慢，越来越复杂，甚至在单体架构应用开发中敏捷模式都无法施展开。这是因为，在单体架构应用中，一个应用中承载的职责太多，其开发、部署和运维复杂度在后期几乎会呈几何性增长，应用的每次编译和启动都需要更长的时间，每次需要修改、增加新的功能时都需要更多的协调和测试。而对于新增的功能，每次修正 Bug 都会使系统的代码更加复杂，使开发-编译-启动-测试进入了一种恶性循环，大大降低了开发效率。

单体架构应用会逐渐变得不稳定，一方面是系统不断增长的复杂性造成的，另一方面是由于系统本身牵一发而动全身的特性造成的，可能一不注意，不常用的模块因为存在内存泄漏而造成整个服务无法正常提供服务，甚至引起服务崩溃。

在数据管理上单体架构应用容易产生漏洞，最常见的就是数据管理。单体架构应用往往要管理的数据类型 / 表都非常多，而且分布在不同团队之间，如果沟通不好可能某些团队的开发会直接操纵数据库表，最常见的方式就是在数据层通过写 SQL 语句操纵数据库表。这种做法相当于埋下不定时炸弹。当数据相关团队修改了数据库结构，由于之前的团队之间使用 SQL 语句进行处理，在系统编译、打包和测试时都有可能通过，但当在真实的生产环境使用时就有可能造成服务的崩溃。

单体架构的应用在开发时要求我们必须使用同一个技术栈，使得单体架构的应用很难接受或切换到其他框架、语言。因为重写一个应用的风险实在太大，即使有更合适的应用开发框架或语言，甚至应用所使用的框架有了新的版本，也很难进行升级。因此，开发者会发现因为这个限制很多应用还在使用着非常古老的技术和框架。

单体架构应用开发对于开发者来说，需要了解更多的东西，如系统架构、统一的开发模式、与之交互的相关模块等，造成了一个新加入的开发者需要更长的时间才能够进入开发状态。

此外，单体架构应用还有一个显著的问题就是由多名开发者同时修改一个代码库。一方面需要开发者自己快速提交所修改的代码，另一方面开发者在每次提交修改时都要心惊胆战地祈求代码不要发生冲突。这也造成了代码库很难快速进入稳定可发布状态。

在部署上，单体架构应用还会造成系统难以进行水平扩展，因为每一个应用实例对服务器来说都需要相同的硬件配置，这让服务器无法充分发挥其能力，造成浪费，并且部署的服务速度会随着代码积累逐渐变慢，性能降低。

1.2　微服务架构

在开发大型、复杂应用时我们常采用的开发方式就是模块化。这种大型应用的开发往往一个人是无法掌控全局的，通过模块化的方式可以将应用分解成多个具有关联的模块，并交由不同开发团队来完成。模块通过开发语言本身的机制进行构建，比如 Java 中我们会打包成一个 JAR 文件，模块之间的调用则是直接使用接口，依赖关系则可以使用 Maven