



This Year in  
**Spring 2018**

**文章精选合集**

一周回顾 发布说明 入门指南 文章精选

# SPRING IN 2018

Spring 技术社区

# 目录

许可证 . . . . .	1
序 . . . . .	2
献辞 . . . . .	3
贡献者 . . . . .	4
引言 . . . . .	6
Spring 年终报告 . . . . .	7
事件回顾 . . . . .	8
年度报告 . . . . .	18
新年展望 . . . . .	37
Spring 一周回顾 . . . . .	38
2018年12月28日 . . . . .	39
2018年12月21日 . . . . .	41
2018年12月14日 . . . . .	43
2018年12月7日 . . . . .	44
2018年11月30日 . . . . .	46
2018年11月23日 . . . . .	47
2018年11月16日 . . . . .	48
2018年11月9日 . . . . .	50
2018年11月2日 . . . . .	51
2018年10月26日 . . . . .	54
2018年10月19日 . . . . .	56
2018年10月12日 . . . . .	57
2018年10月5日 . . . . .	58
2018年9月28日 . . . . .	60
2018年9月21日 . . . . .	61
2018年9月14日 . . . . .	63
2018年9月7日 . . . . .	65
2018年8月31日 . . . . .	66
2018年8月24日 . . . . .	67
2018年8月17日 . . . . .	68
2018年8月10日 . . . . .	69
2018年8月3日 . . . . .	71
2018年7月27日 . . . . .	73
2018年7月20日 . . . . .	74
2018年7月13日 . . . . .	75
2018年7月6日 . . . . .	76
Spring Framework 发布说明 . . . . .	77
Spring Framework 5.1.3 发布 . . . . .	79
Spring Framework 5.1.2 发布 . . . . .	80
Spring Framework 5.1.1 发布 . . . . .	81
Spring Framework 5.1.0 发布 . . . . .	82
Spring Framework 5.1 RC3 发布 . . . . .	83

Spring Framework 5.0.11 发布	84
Spring Framework 5.0.10 发布	85
Spring Framework 5.0.9 发布	86
Spring Framework 4.3.21 发布	87
Spring Framework 4.3.20 发布	88
Spring Framework 4.3.19 发布	89
Spring Boot 发布说明	90
Spring Boot 2.1.1 发布	92
Spring Boot 2.1.0 正式发布	93
Spring Boot 2.0.7 发布	95
Spring Boot 2.0.6 发布	96
Spring Boot 2.0.5 发布	97
Spring Boot 2.0.4 发布	99
Spring Boot 1.5.18 发布	100
Spring Boot 1.5.17 发布	101
Spring Boot 1.5.16 发布	102
Spring Boot 1.5.15 发布	103
Spring Data 发布说明	104
Spring Data Lovelace SR3 发布	106
Spring Data Lovelace SR2 发布	107
Spring Data Lovelace SR1 发布	108
Spring Data Lovelace 正式发布	109
Spring Data Kay SR12 发布	111
Spring Data Kay SR11 发布	112
Spring Data Kay SR10 发布	113
Spring Data Ingalls SR17 发布	114
Spring Data Ingalls SR16 发布	115
Spring Data Ingalls SR15 发布	116
Spring Security 发布说明	117
Spring Security 5.1.2 发布	119
Spring Security 5.1.1 发布	120
Spring Security 5.1.0 发布	121
Spring Security 5.1.0.RC2 发布	122
Spring Security 5.1.0.RC1 发布	125
Spring Security 5.0.10 发布	128
Spring Security 5.0.9 发布	129
Spring Security 5.0.8 发布	130
Spring Security 4.2.10 发布	131
Spring Security 4.2.9 发布	132
Spring Security 4.2.8 发布	133
Spring Cloud 发布说明	134
Spring Cloud Greenwich.RC2 发布	137
Spring Cloud Greenwich.RC1 发布	141
Spring Cloud Greenwich.M3 发布	146
Spring Cloud Finchley.SR2 发布	150

Spring Cloud Edgware.SR5 发布	154
Spring Cloud for Alibaba 0.2.1 发布	157
Spring Cloud for Alibaba 0.2.0 发布	159
Spring Cloud Data Flow 1.7 正式发布	161
Spring Cloud Data Flow 1.6 正式版发布	164
Spring Cloud Stream Fishtown.RC2 /2.1.0.RC2.发布	165
Spring Batch 发布说明	166
Spring Batch 4.1.0 发布	168
Spring Batch 4.1.0.RC1 发布	178
Spring Batch 4.1.0.M3.发布	179
Spring Integration 发布说明	181
Spring Integration 5.1.0 发布	183
Spring Integration、AMQP和Kafka RC1.发布	187
Spring Integration for AWS 2.0 发布	188
Spring 开发指南	192
Spring开发指南，带你进入Spring的世界	194
安装 Java SDK.8	196
安装 Maven	197
安装 Spring Boot CLI	198
Spring Boot CLI 使用详解	200
安装 STS	208
Spring 精选文章	211
最全 Spring 书单，送给爱学习的你	213
Spring Boot 最佳实践	217
10 种保护Spring Boot应用程序的绝佳方法	222
介绍 Spring Cloud Function.项目	231
介绍 Spring Data JDBC	234
Spring Cloud Function 对 Kotlin.语言的支持	239
参考资料	240

# 许可证

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit<http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

# 序

Spring 最早可以追溯到 Rod Johnson 写的第一本书《Expert One-on-One J2EE Design and Development》，书中涉及到的 3 万行代码，形成了最早期的 Spring 框架核心的原型（interface21）。随后，Rod Johnson 与 Juergen Hoeller 一起主导开发并开源了 Spring Framework。经过一番努力，2014 年 3 月，Spring Framework 正式发布 1.0 版本。

现在是 2018 年，距离第一次版本发布已经过去 15 年了，但是 Spring 仍然是目前最流行的 Java 企业级应用开发框架，同时也已经突破了最初提供的功能，现如今已经发展成为一个使用和围绕 Spring 框架而构建起来的整个项目家族。Spring 家族项目解决广泛的技术问题，支持使用 Spring 功能构建整体解决方案。从 Web 应用，数据存储和批处理技术，企业应用集成，安全，微服务，云计算等等都有相应的框架和技术提供支持。

Spring Framework 5 为开发者带来了新的响应式 Web 技术栈，为异步非阻塞场景提供了技术支持。Spring Boot 2.0 提供了众多可供开发者选择的起步依赖，让各个框架和第三方集成变得从未有过的简单，几乎不需要配置或者少量必要配置就可以开发者构建一个新的应用，可以开发 REST API，Web Socket，也可以是任务调度，流计算和批处理等。

Spring Cloud 使得构建分布式系统变得容易并且不再出错。Spring Cloud 为最常见的分布式系统模式提供了简单易用的编程模型，帮助开发人员构建弹性、可靠和协调的应用程序。Spring Cloud 构建于 Spring Boot 之上，使开发人员可以轻松入门并快速提高工作效率。

Spring Cloud Data Flow 是一个编排工具，为最常见的集成方案提供开箱即用的连接器，使连接系统变得简单，可以轻松构建和编排用于数据收集，实时分析和数据导入/导出等用例的云原生数据管道。

综上所述，Spring Boot 帮助开发者快速构建应用，Spring Cloud 用于构建分布式系统，协调所有的服务，而 Spring Cloud Data Flow 就是连接这些服务之间的数据管道，它们共同组成了 Spring 项目家族的三驾马车，共同为开发者构建强大的系统而保驾护航。

时代在变，技术在变，对技术的追求也永无止境，我们都永远在路上！

# 献辞

感谢 Rod Johnson，是他创造了 Spring Framework，并开放源码，让全世界的 Spring 开发者都享受到了这个框架带来的好处！

感谢 Phillip Webb，是他创造了 Spring Boot，让 Spring 开源项目有了新的生命和活力，再次点燃了所有 Spring 开发者的热情！

感谢 Matt Raible，他是我开源和职业生涯的启蒙导师，教会了我很多在学校学不到的东西，追随他的脚步一路走来，收益良多，期待他能来得中国举办一场技术分享会！

感谢 Josh Long，他是 Spring 最受欢迎的技术布道师，他让整个 Spring 开发充满了乐趣，他坚持写了 8 年的《This week in Spring》，让所有人都收益良多！我就是其中一位忠实的粉丝！

感谢 Spring 开源和技术社区，以及所有的 Spring 开发者，让技术分享和传播变得有价值！

感谢 Apache 软件基金会，作为专门为支持开源软件项目而办的一个非盈利性组织，它使得开源精神发扬光大，它所支持的开源项目，让全世界从事软件和编码的人都收益匪浅！

最后，感谢您！是您选择了 Spring，选择了美好，选择了阅读这本书，希望也分享给您的朋友们！

# 贡献者

这本书以电子版的形式发布，首先得感谢 Pivotal 对 Spring 开源的支持，以及所有 Spring 项目负责人、开发者、代码和问题提交者等等参与到开源的所有人的努力。下面是 Spring 官方博客上编写和发布文章的所有人员。

Andy Wilkinson

Artem Bilan

Brian Clozel

Christoph Strobl

Craig Walls

Dave Syer

Diógenes Rettori

Filip Hanik

Gary Russell

Glenn Renfro

Greg Turnquist

Gunnar Hillert

Jakub Pilimon

Jens Schauder

Joe Grandja

John Blum

Jon Schneider

Josh Cummings

Josh Long

Juergen Hoeller

Madhura Bhave

Mahmoud Ben Hassine

Marcin Grzejszczak

Mark Heckler

Mark Paluch

Mark Pollack

Martin Lippert

Matt Raible

Michael Minella

Nieraj Singh

Oleg Zhurakousky

Oliver Drotbohm

Phil Webb

## 贡献者

Pieter Humphrey

Rob Winch

Rossen Stoyanchev

Roy Clarkson

Ryan Baxter

Scott Frederick

Sébastien Deleuze

Simon Baslé

Soby Chacko

Spencer Gibb

Stephane Maldini

Stéphane Nicoll

Thomas Risberg

Xiao Jing

# 引言

很感谢，你将花费你生命中的若干小时来阅读我为你在 2018 年底精心准备的有关 Spring 的相关内容。让我用几分钟时间来介绍下本书内容。

在**Spring 年度报告**，我们将从总体上回顾 2018 年发生的一些重要事件，从开源社区和组织，到软件企业、商业巨头、云厂商。接着，重点分析了 Spring 几个主要项目的一些度量数据，如项目活跃度、贡献者排行、版本发布情况。最后是接下来新的一年的思考和展望。

在**Spring 一周回顾**，这是 2018 年度下半年每周五发布的一周事件回顾，主要包括 Spring 开源项目的发布说明，技术社区举办的一些活动，另外也包括 Apache 开源项目、JBoss 开源项目、前端、容器、开源工具等最新的资讯报道。

接下来七个章节，依次介绍了 Spring Framework、Spring Boot、Spring Data、Spring Security、Spring Cloud、Spring Batch、Spring Integration 等框架的各个主要版本的发布说明，便于技术选型和使用指导。

**Spring 开发指南**介绍了开发指南中，一些流行工具的安装，重点介绍了官方开发指南、快速入门、专题相关内容。

**Spring 精选文章**包括了 2018 年，翻译的一些有关 Spring Boot、Spring Security、Spring Cloud 等项目的介绍，以及最佳实践，同时提供了近 10 年来，有关 Spring 的技术书籍清单，方便大家学习。

下面让我们开始。

# Spring 年终报告

2018 年注定是一个不平凡的年份，对于开源（Open Source）来说是诞生二十年的幸运之年，这一年开源界也发生了太多的重大事件，值得我们好好思考和总结。开源已经并继续影响着每一个开发者，每一家公司，无论是谁都是这场开源运动的参与者。

2018 年也注定是收获和成长的一年，国内各大公司也都在积极参与开源生态建设和推广，这只是刚刚开始，未来还有很长的路要走。

Spring 技术社区，只是其中的一个技术社区，但是 Spring 自从诞生之初就埋下了种子，Spring 不仅仅是开源技术，更是一直开源精神和文化，早已深入人心，并赢得了开发者的信赖。作为 Spring 的拥有公司 Pivotal 在 Spring 开源技术上始终保持初心，持续的人员和研发投入，Spring Boot 和 Spring Cloud 继续在微服务技术领域保持领先优势，在今年9月底成功举办了最大的 Spring 技术盛会 SpringOne Platform 2018，接着 Spring Tour 又在11月3号在北京举办，这给所有 Spring 开发者提供了技术学习和交流的机会，感谢他们。

时代在进步和发展，技术也在变革，也永远不会停止前进的步伐，而我们唯一需要做的就是不断学习、参与其中。

## 事件回顾

先来回顾下这一年都发生了哪些大事件。

### 开源 20 周年

**开源 (Open Source)** 20 周年，各地相继举办各种庆祝活动。

1998 年 2 月 3 日，“开源”这一术语率先被用于软件领域。之后不久，开源的定义被创建和确认，开放源代码促进会 OSI 开始发起“开源运动”对其广泛传播。

二十年后，这一运动被证明是非常成功的，取得了超乎当时所有人的想象。如今，开源软件无处不在，并成为互联网和 Web 的基础，为我们日常使用的电脑和移动设备，以及它们所连接的网络提供“动力”。没有它，云计算和新兴的物联网不会拓展如此之快，甚至都不一定会被创造出来。它带来并验证了新的商业模式，允许像 Google 和 Facebook 这样的大公司再攀高峰。但同时，它也有一个“黑暗”的一面，为犯罪分子提供了新的诈骗和攻击手段，并带来专利控制问题。

经过二十年的发展，开源已趋于成熟。如果说第一个十年，是倡导和争议的十年，那第二个十年，则是采纳和适应的十年。

1998~2008	2008~2018
第一个十年，大家的重点围绕在商业模式上 - “要怎么样让别人自由使用的同时而得到报酬”？	第二个十年，出现了新的问题 - “我要怎么参与开源”？
第一个十年，开源项目主要是在替代现成的产品；	第二个十年，开始逐渐成为解决方案的组成部分。
第一个十年，项目往往由非正式的个人组织进行；	第二个十年，更多的项目被企业或基金会构建和经营。
第一个十年，开发者往往只投入单一的项目，并利用业余时间维护开发；	第二个十年，越来越多的开发者被雇佣去专门开发和维护开源项目。
第一个十年，开源软件与“自由软件”产生了冲突；	第二个十年，随着开源软件的快速发展，冲突基本已被忽略。

那么，第三个十年，开源软件将何去何从？

- 新商业模式 — 如何整合大量开源项目从而形成更复杂的解决方案，将成为新的不可忽视的商业模式，尤其是在部署和扩展方面。
- 更实用 — 开源项目将成为各种解决方案的主要组成部分。
- 更“抱团” — 越来越多的项目将由诸如 Linux 基金会、Apache 基金会等机构主导。
- 更专业/更通用 — 开发者将更多地被要求将许多开源技术集成到复杂的解决方案中，并为一系列项目做出贡献。
- 更自由 — 随着新问题的出现，软件将会更自由地被用于针对协作社区和独立部署人员的解决方案中。

### Pivotal 在纽交所上市

1989 年由 Pivotal 现任 CEO Rob Mee 创立的咨询公司 Pivotal Labs，专注于快速的互联网式软件开发，即敏捷编程。

2012 年，Pivotal Labs 被 EMC 收购。

2013 年，EMC 和 VMware 分拆出其 Cloud Foundry、Pivotal Labs、Greenplum 等云计算、大数据资源，GE 投资 1.05 亿美元，成立新公司 Pivotal。

2014年，Pivotal 成立 Cloud Foundry 基金会，致力于推动建立 PaaS 全球标准。

2015年到2016年，Pivotal 陆续实现了三大公有云厂商 AWS、Azure 和 GCP 对 PCF 的支持，进一步拓展了业务生态。

2016年还发生了一件大事，DELL 收购了 EMC，由此成为 Pivotal 控股股东，为 Pivotal 带来了更多的战略资源。

2017年，埃森哲与 Pivotal 共同组建 Accenture Pivotal Business Group，由埃森哲负责运营和管理，专注于帮助大企业将业务快速迁移到 PCF 上。随着容器编排工具 Kubernetes 的火热，Pivotal 与 VMware、Google 联合推出了基于 Kubernetes 的容器服务 PKS。

2018 年 4 月 20 日，Pivotal 于纽约证券交易所上市。截至 2018 年 6 月 22 日收盘，股价较 IPO 发行价累计上涨 67.7%，市值达到 64.7 亿美元，Pivotal 用业绩赢得了资本市场的认可。

## Microsoft 收购 GitHub

官方报道：2018年6月4日， Microsoft 宣布 75 亿美元收购 GitHub



图 1. 左起：Chris Wanstrath，GitHub 首席执行官兼联合创始人；微软首席执行官 Satya Nadella；和微软公司负责开发人员服务的副总裁 Nat Friedman

目前已超过 2800 万的开发者在 GitHub 上进行协作开发，并托管了超过 8500 万个代码库。微软是一家以开发者为中心的公司，构建技术与他人共享是其使命和核心价值观。GitHub 作为开发者学习、共享和成长的基地，也正是微软的“目的地”。

完成收购后，GitHub 将仍然是一个开放平台，并保留其开发者优先的风格，独立运营。

## Java 11 正式发布

Java 11 于 2018 年 9 月 26 日正式发布！。

JDK 11 将是一个企业不可忽视的版本，首先很重要的是安全更新、新特性、不断改进和优化的 JVM，其次，11 是一个长期支持版本（LTS）。对于企业来说，选择 11 将意味着长期的、可靠的、可预测的技术路线图。

## SpringOne Platform 2018

### SpringOne Platform 2018

在 9 月 24-27 日 为期四天的 SpringOne Platform 2018 大会上，来自业界的技术大咖们分享了最新的技术资讯、最佳实践和行业案例介绍。官方也已经在这几天在 YouTube 上放出了全部视频：SpringOne Platform 2018 视频清单。

详细内容报道：

- SpringOne Platform 2018

### Spring Tour 2018 北京站

11月3日，Spring 开源项目背后的公司 Pivotal 在北京举办 SpringOne Tour 技术峰会，包括 Pivotal 的 Spring 技术布道师 Josh Long、Mark Heckler、Spencer Gibb、Paul Czarkowski 等在一众 Spring 技术大咖来到中国，还有 Pivotal 中国的资深架构师和专家刘凡、李刚一起为广大 Spring 开发者们献上一道 Spring 技术盛宴。现场总共到了将近 500 多位 Spring 技术爱好者，和大咖们一起讨论技术问题、合影留恋，这是一次非常棒的技术之旅。

详细内容报道：

- Spring Tour 2018 北京



开源社举办开源年会 2018



图 2. 2018 开源社开源年会于10月20-21日在深圳举办



图 3. 开源社庆祝开源 20 周年



图 4. 2018 开源年度报告

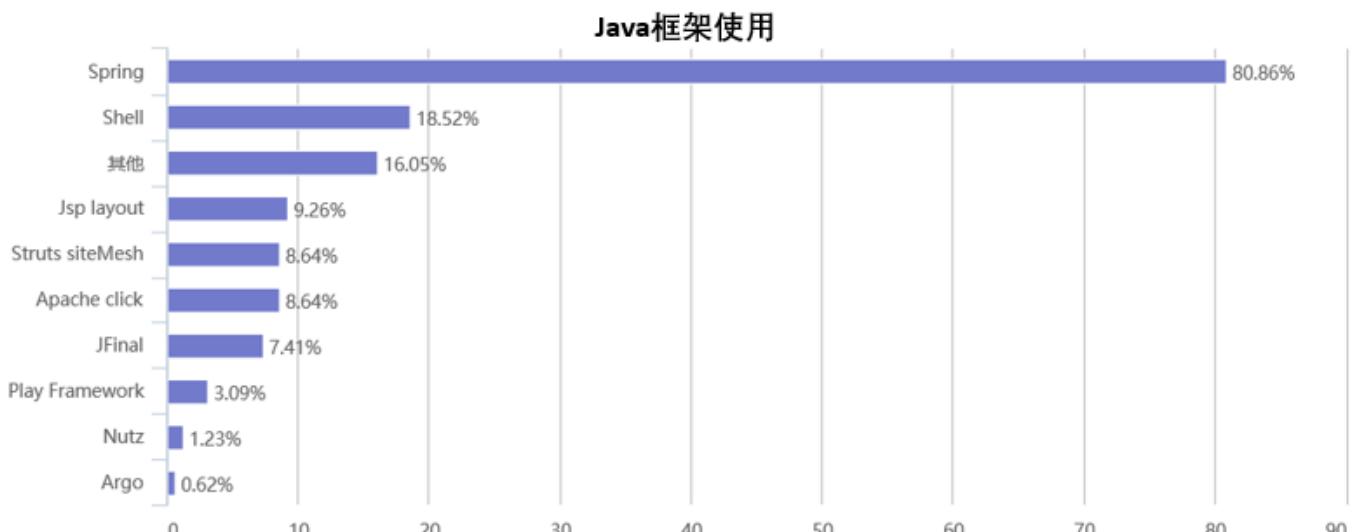


图 5. Java 框架使用报告显示 Spring 最高

[在线阅读报告](#)

详细内容报道：[开源社 开源年会 2018](#)

## 开源中国源创会 2018

2018年，开源诞生20年，也是开源中国成立的十周年，今年的年终盛典在深圳举办，以“与十俱进，码出未来”为主题。



图 6. 开源中国源创会 2018 在深圳

## IBM 收购 RedHat



图 7. IBM 340 亿美元收购 RedHat

[官方详细报道](#)

## Spring Cloud Netflix 模块进入维护状态

Netflix 旗下 Eureka、Hystrix、Ribbon 等开源项目相继停止更新，Spring Cloud Netflix 模块相应也在年底宣布进入维护模式。

[详细内容报道：Spring Cloud Greenwich.RC1 发布](#)

## 阿里巴巴开源

Alibaba 高调宣布宣布 Nacos 和 Setinal 开源，同时阿里巴巴开源和阿里云中间件产品，开始与 Spring Cloud 进行深度集成，进入 Spring Cloud 孵化项目！目前已发布 0.2.1，预计将在 2019 年 2 月份随 H 版本发布。

相关内容：

- [Spring Cloud for Alibaba 0.2.1 发布](#)
- [Spring Cloud for Alibaba 0.2.0 发布](#)

## Micronaut 开源

在 Java 领域，开源的框架一直都非常多，从 WebWork、Struts、Tapestry、Grails、Dropwizard、Ratpack、Spark，包括 Spring Boot，这些都是非常优秀的开源框架。Micronaut 被大多数 Spring & Grails 的开发者寄予厚望，Micronaut 于今年 5 月 23 日宣布 [正式开源](#)之后，开发团队一直非常努力投入开发之中，在接连发布了 M1、M2、M3、M4 以及 RC1、RC2 等版本之后，在 10 月 23 日，正式发布 1.0 GA！Micronaut 的开发团队来自 OCI 公司，大部分人员都是 Spring 和 Grails 开发人员，对 Spring 优缺点非常了解。

Micronaut 包含现代开发人员为 JVM 高效构建微服务所需的所有工具，而不会影响内存占用和启动时间，包括：

- 集成的编译时依赖注入和 AOP
- 云原生配置管理
- 服务发现和客户端负载均衡
- 集成的 HTTP 客户端和基于服务器的 Netty
- 合理的默认值和自动配置

详细内容报道：[Micronaut 版本发布说明](#)

## 服务网格

Service Mesh 2018 年已经很火热，2019 年会不会更加火爆？Spring Cloud 与 Service Mesh 关系如何？

Spring Cloud 目前有个子项目 [spring-cloud-kubernetes](#)，当前版本是 1.0.0.RC2，正式版本将会随 [Greenwich](#) 正式版本一起发布。目前主要集成点是：服务发现和配置管理。2019 年又会怎样发展下去，持续关注。

延伸阅读：

- [Kong 1.0 正式发布！](#)

# 年度报告

Spring Framework 目前仍然保持着“一年一个小版本，四年一个大版本”的迭代速度继续前进。**5.0** 版本将会随着 Spring Boot 2.0 一起终止支持（大概是 2019 年 3 月底）。**5.1** 版本是一个过渡性版本，2019 年底将会不再支持。**5.2** 版本将会是一个长期支持版本（LTS）版本，可能支持到 2023 年底。

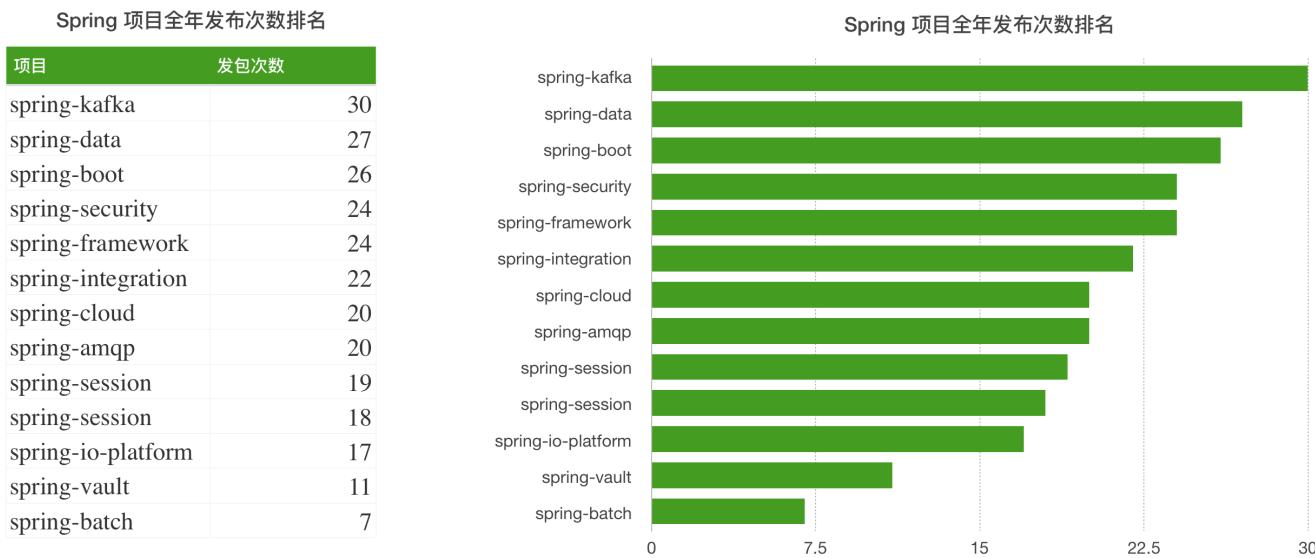


图 8. 2018 年项目发布次数排行榜

表格 1. 2018 年项目发布次数清单

序号	项目	发布次数
1	spring-kafka	30
2	spring-data	27
3	spring-boot	26
4	spring-security	24
5	spring-framework	24
6	spring-integration	22
7	spring-cloud	20
8	spring-amqp	20
9	spring-session	19
10	spring-statemachine	18
11	spring-io-platform	17
12	spring-vault	11
13	spring-batch	7

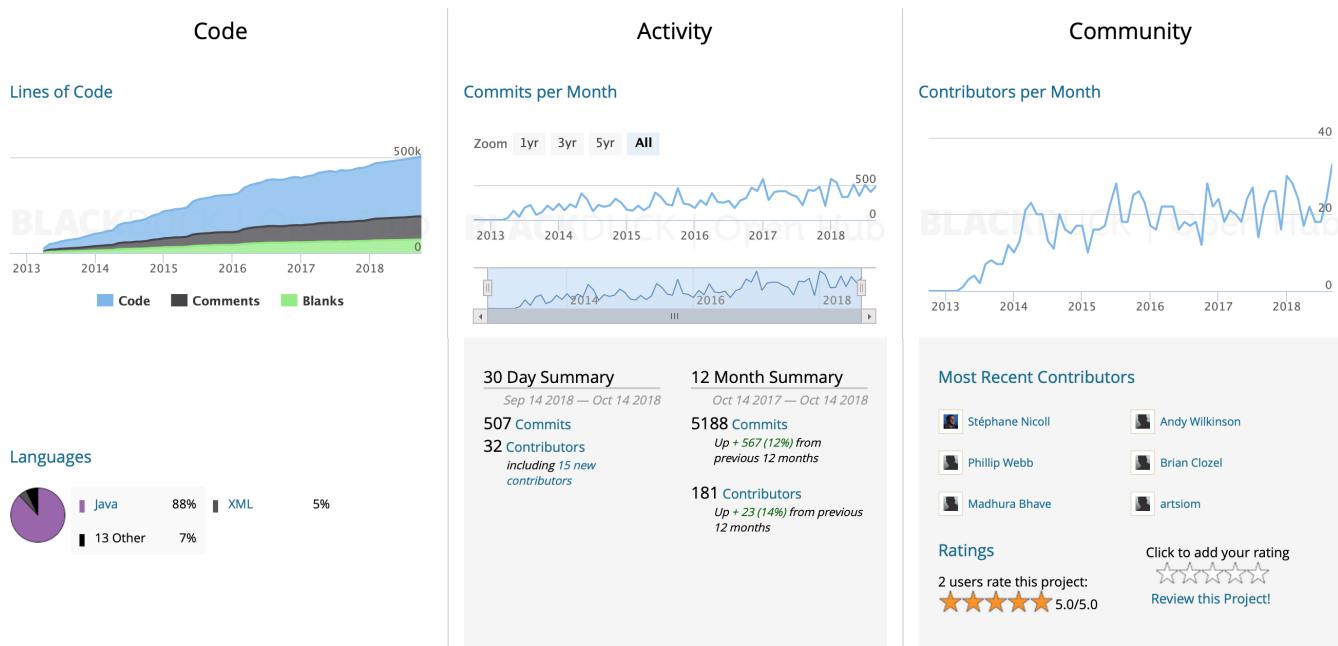
序号	项目	发布次数
14	spring-ws	5
15	spring-webflow	4
16	spring-restdocs	4
17	spring-credhub	3
18	spring-mobile	3

## Spring Boot 项目

### 主要版本发布时间

版本	发布时间
2.1.0	2018-10-30 10:50:21
2.0.0	2018-03-01 04:42:15
1.5.0	2017-01-30 11:40:15
1.4.0	2016-07-28 19:53:52
1.3.0	2015-11-16 11:18:34
1.2.0	2014-12-11 00:35:35
1.1.0	2014-06-10 19:40:29
1.0.0	2014-04-01 10:05:45

### 项目活跃度

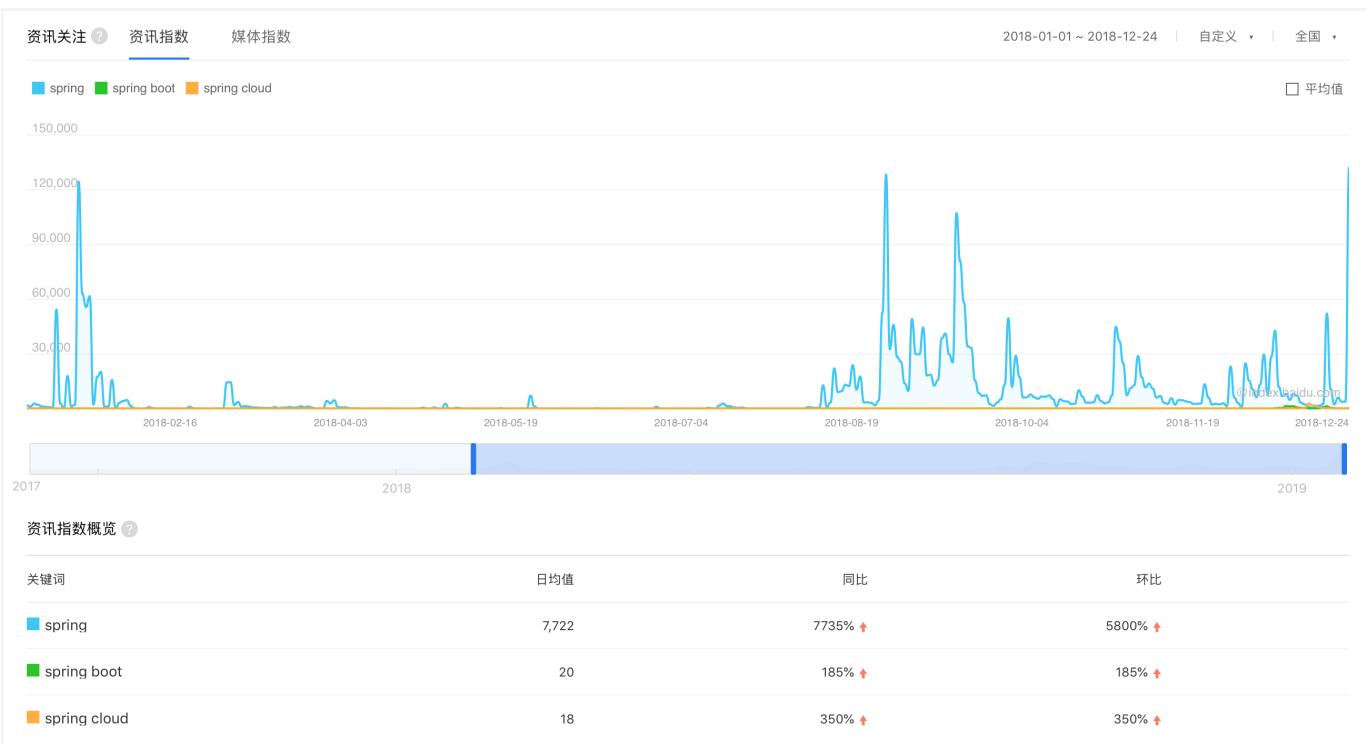


## 项目贡献者

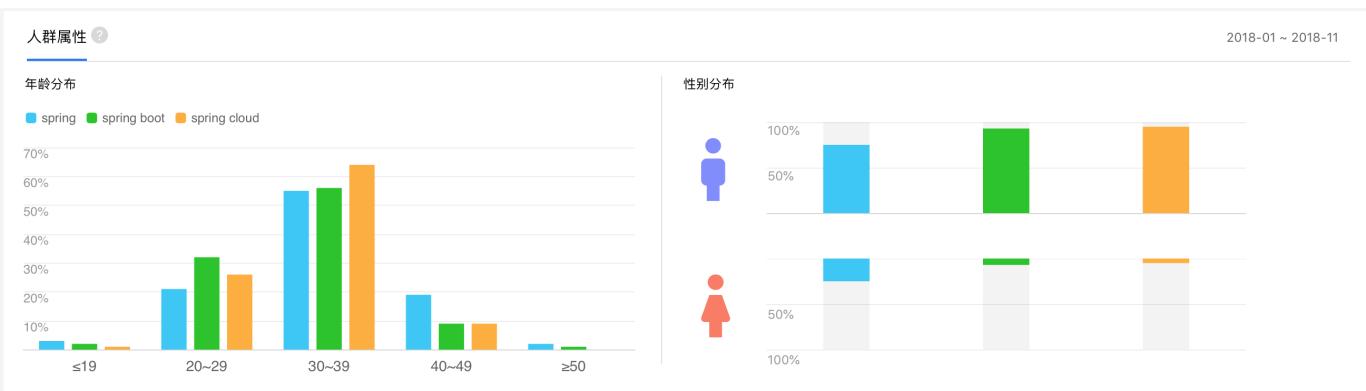
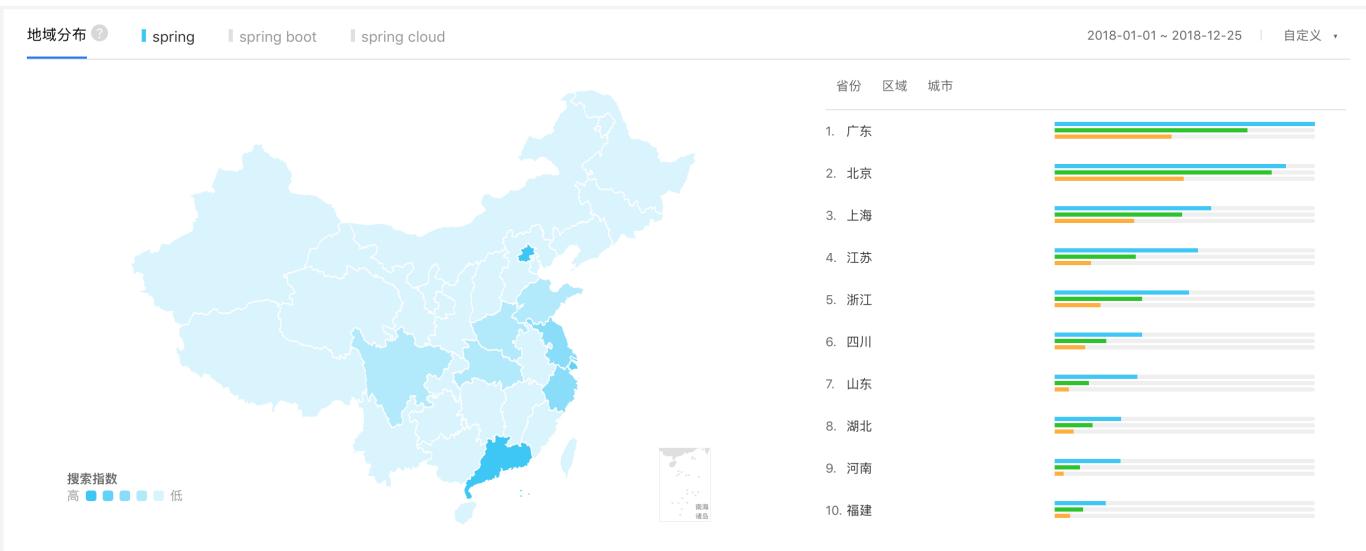
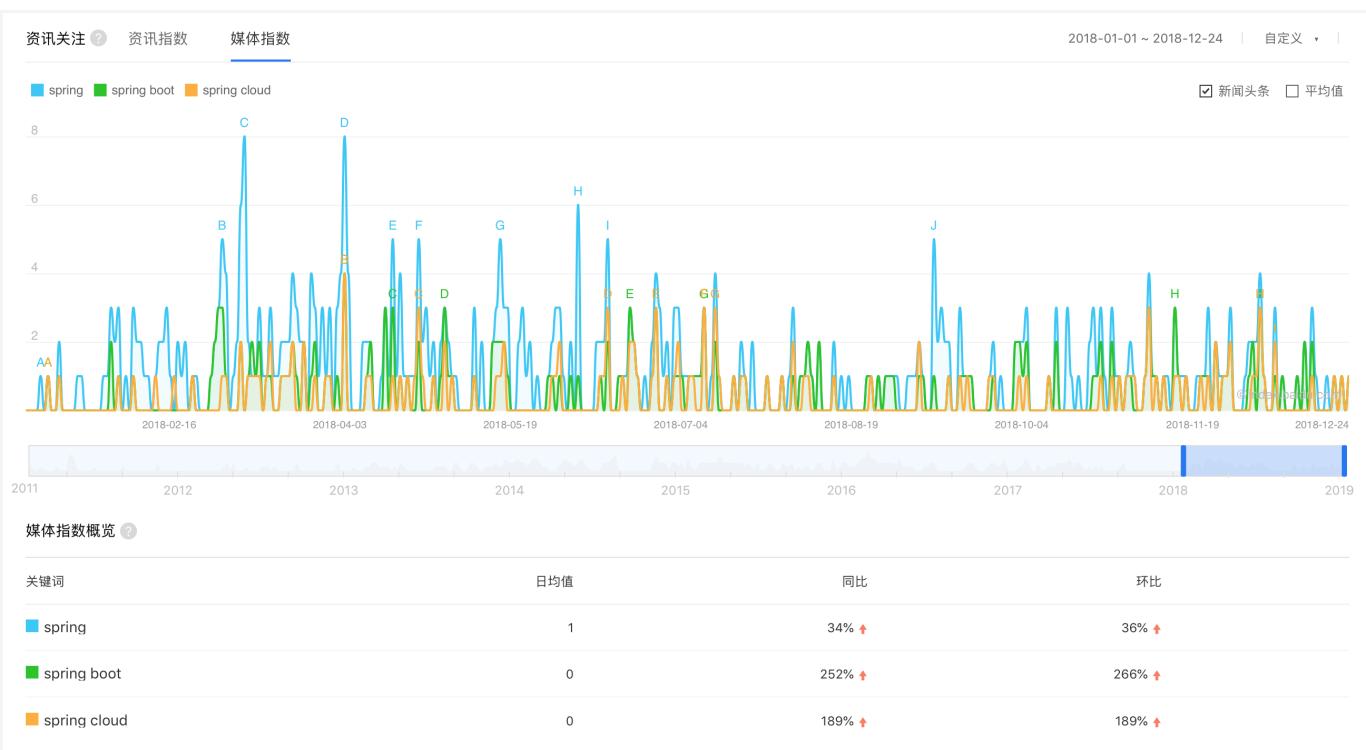


## 搜索指数

# Spring 年终报告



# Spring in 2018



## 版本使用调查

### 结果分析：

1. Spring Boot 总共有 49 票，目前使用最多的版本是 1.5.x。

2. 大部分公司从 Spring Boot 1.3、1.4 版本开始使用，升级到 1.5 版本是比较稳妥的，1.5 版本也相对稳定，技术资料和书籍相对较多，同时生产环境部署也以 1.5 版本居多，不太轻易会升级到 2.0。

Spring Boot 使用版本调查结果

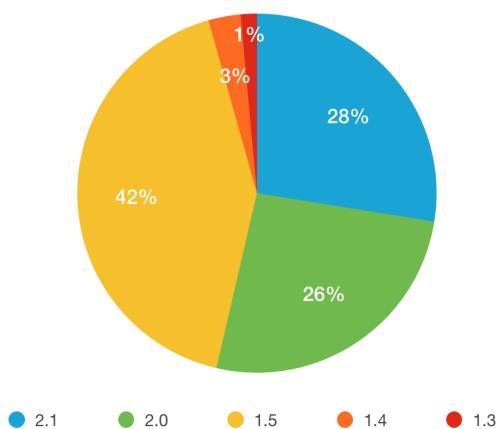
版本	投票数
2.1	19
2.0	18
1.5	29
1.4	2
1.3	1

#### 结果分析：

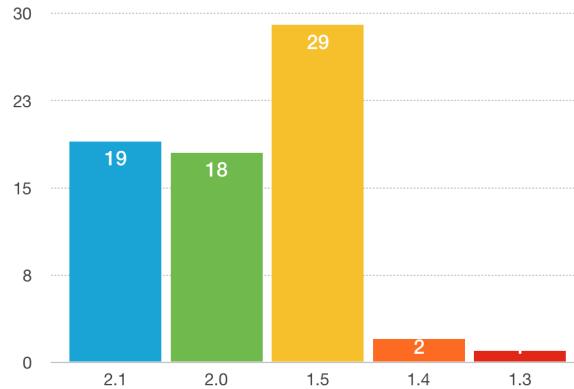
1. Spring Boot 总共有 51 票，目前使用最多的版本是 1.5.x。

2. 大部分公司从 Spring Boot 1.3、1.4 版本开始使用，升级到 1.5 版本是比较稳妥的，1.5 版本也相对稳定，技术资料和书籍相对较多，同时生产环境部署也以 1.5 版本居多，不太轻易会升级到 2.0。

Spring Boot 使用版本



Spring Boot 使用版本



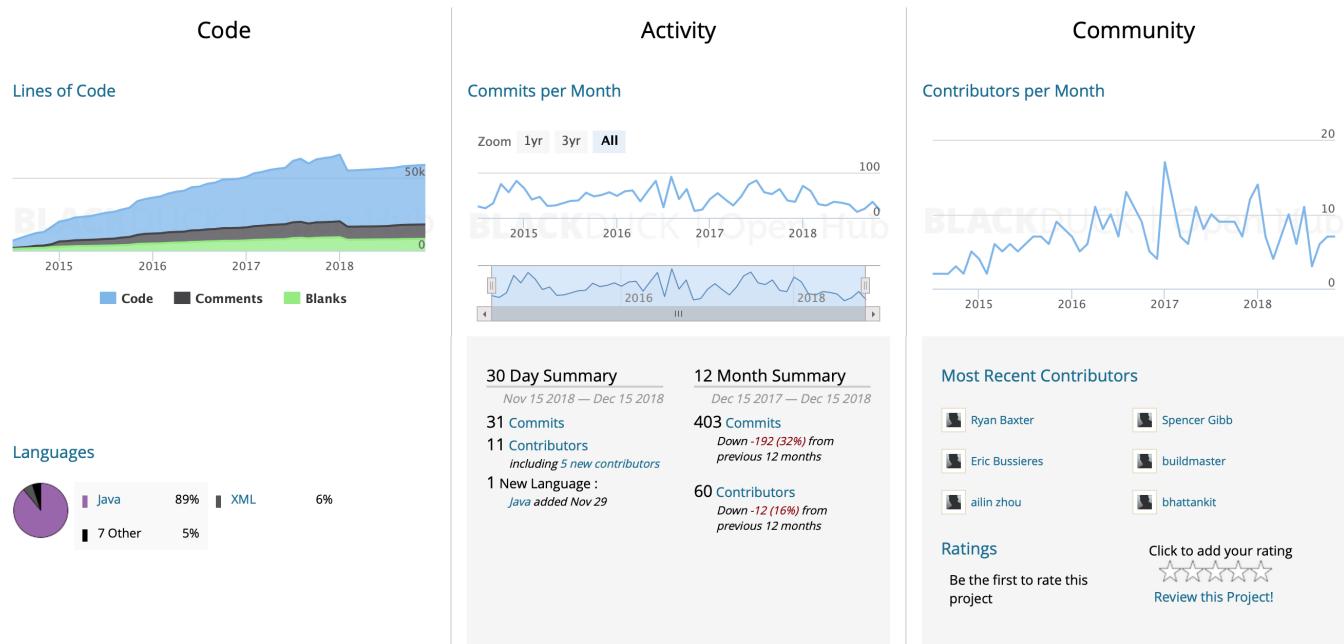
## Spring Cloud 项目

### 主要版本发布时间

版本	发布时间
Greenwich	2019-01-17 00:00:00
Finchley	2018-06-19 01:57:34
Edgware	2017-11-21 23:24:10
Dalston	2017-04-12 17:15:14
Camden	2016-09-25 06:42:34
Brixton	2016-05-11 16:26:20
Angel	2015-06-23 10:14:49
1.0.0	2015-03-03 14:14:55

版本	发布时间
1.0.0.M1	2014-10-07 12:43:27

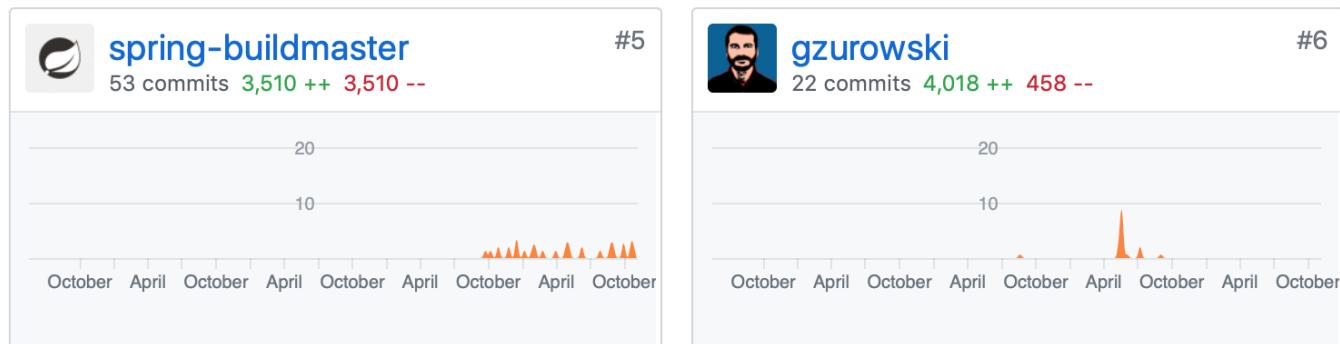
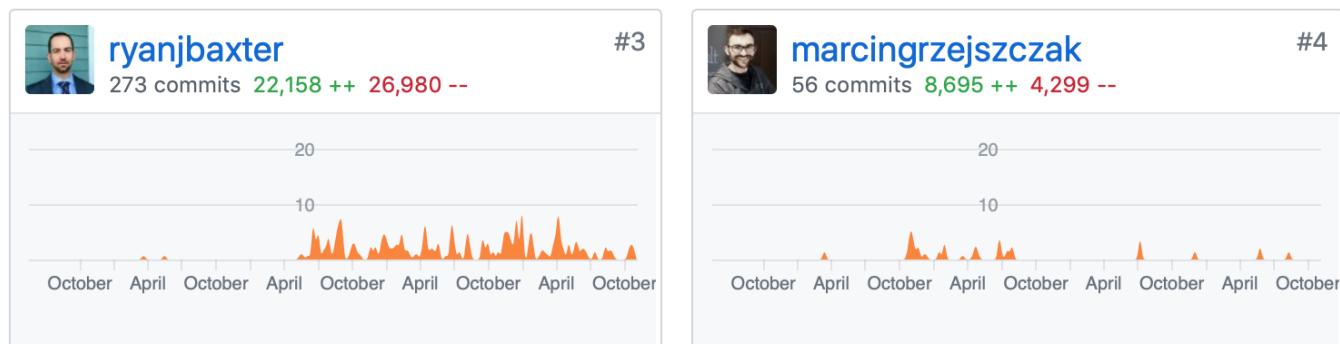
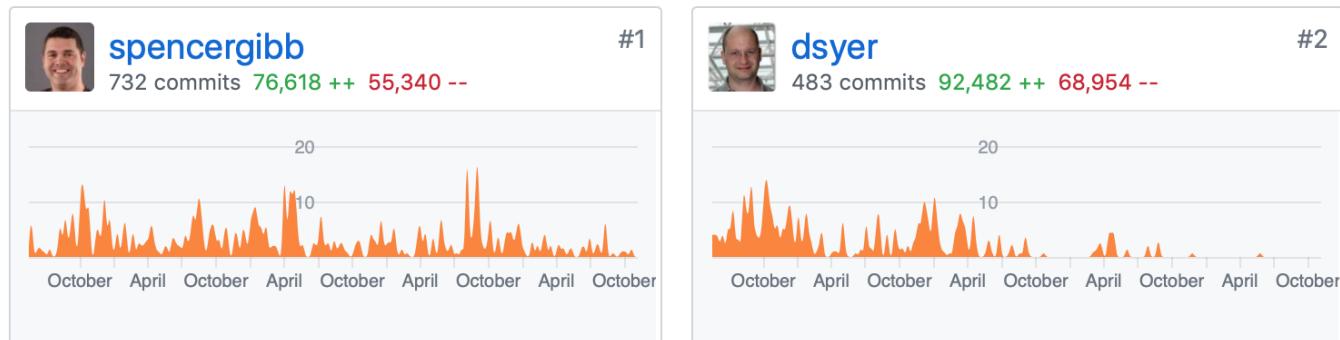
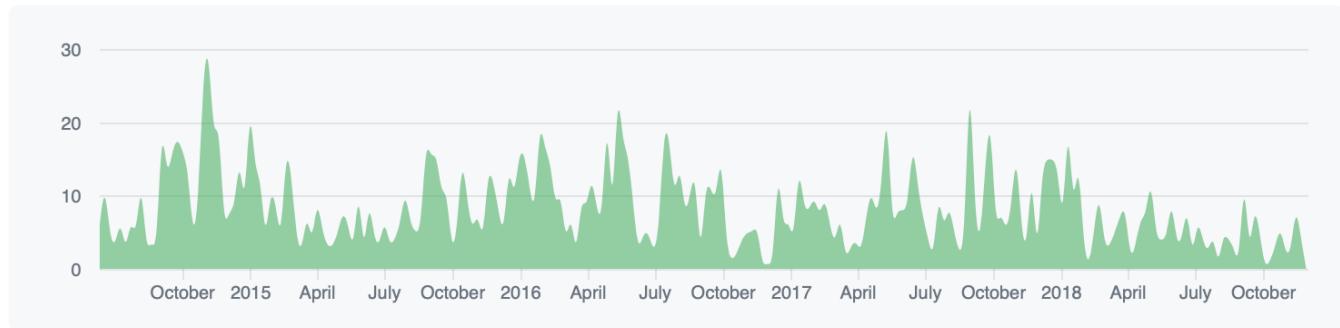
## 项目活跃度



## 项目贡献者

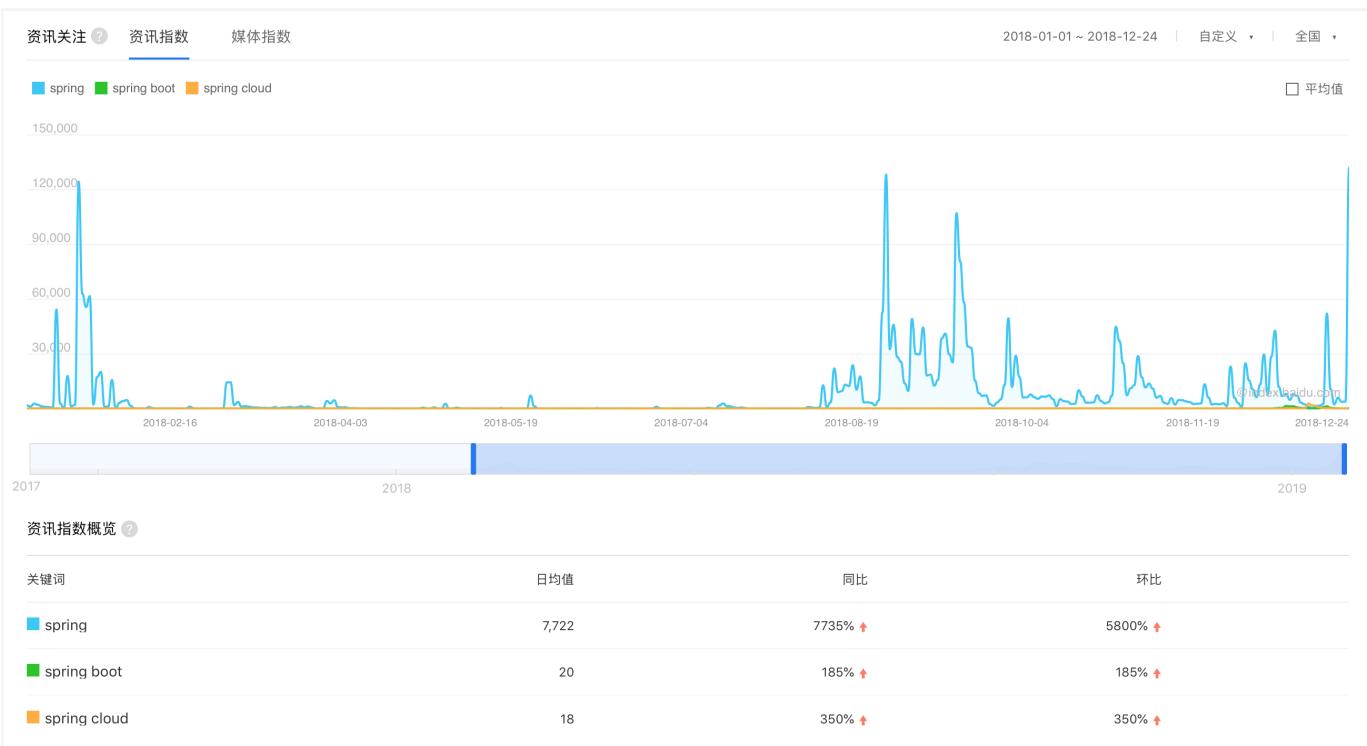
# Spring 年终报告

Contributions to master, excluding merge commits

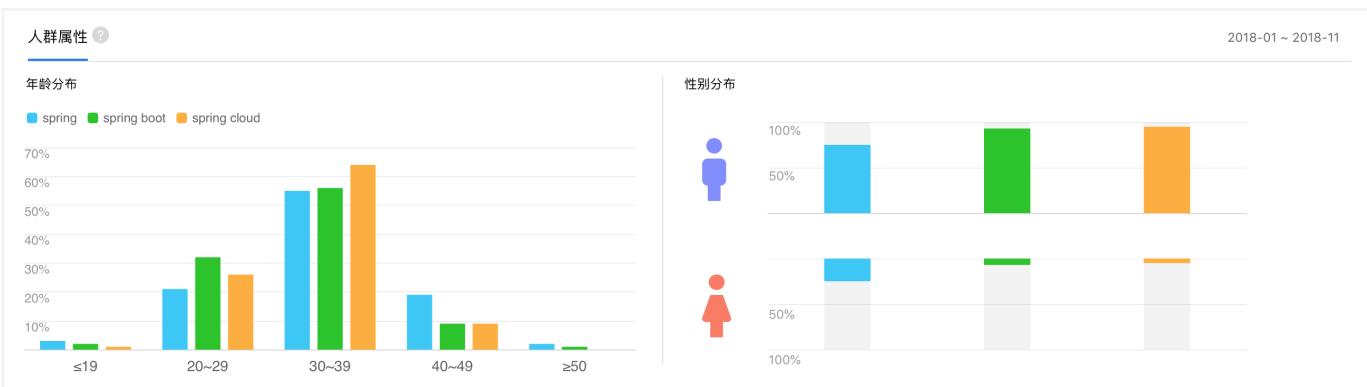
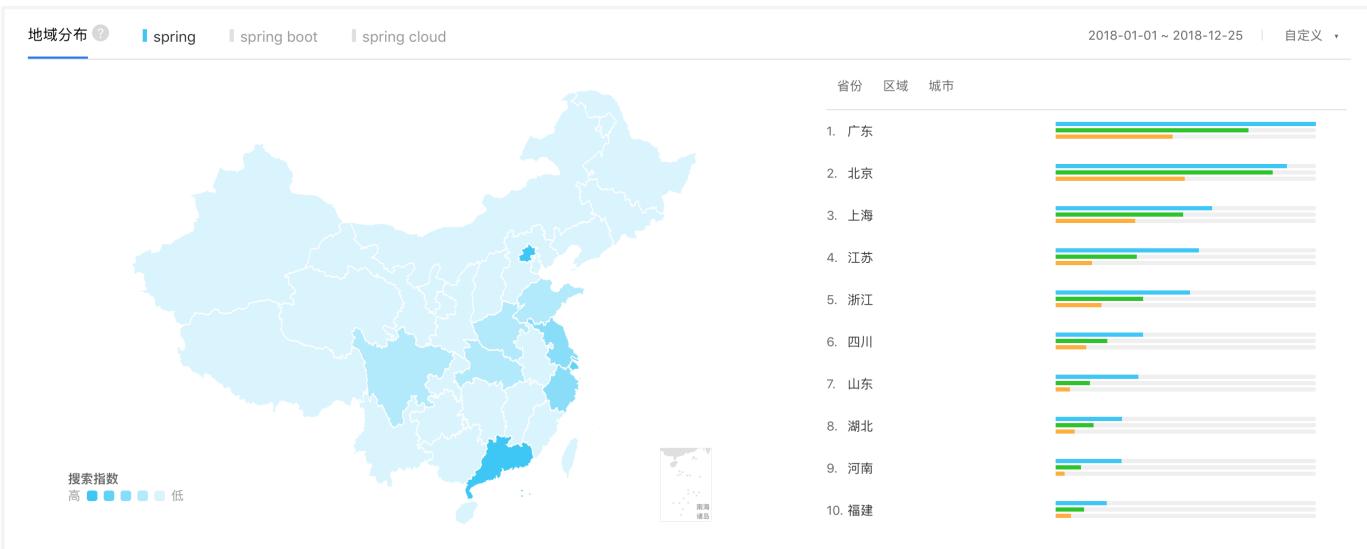
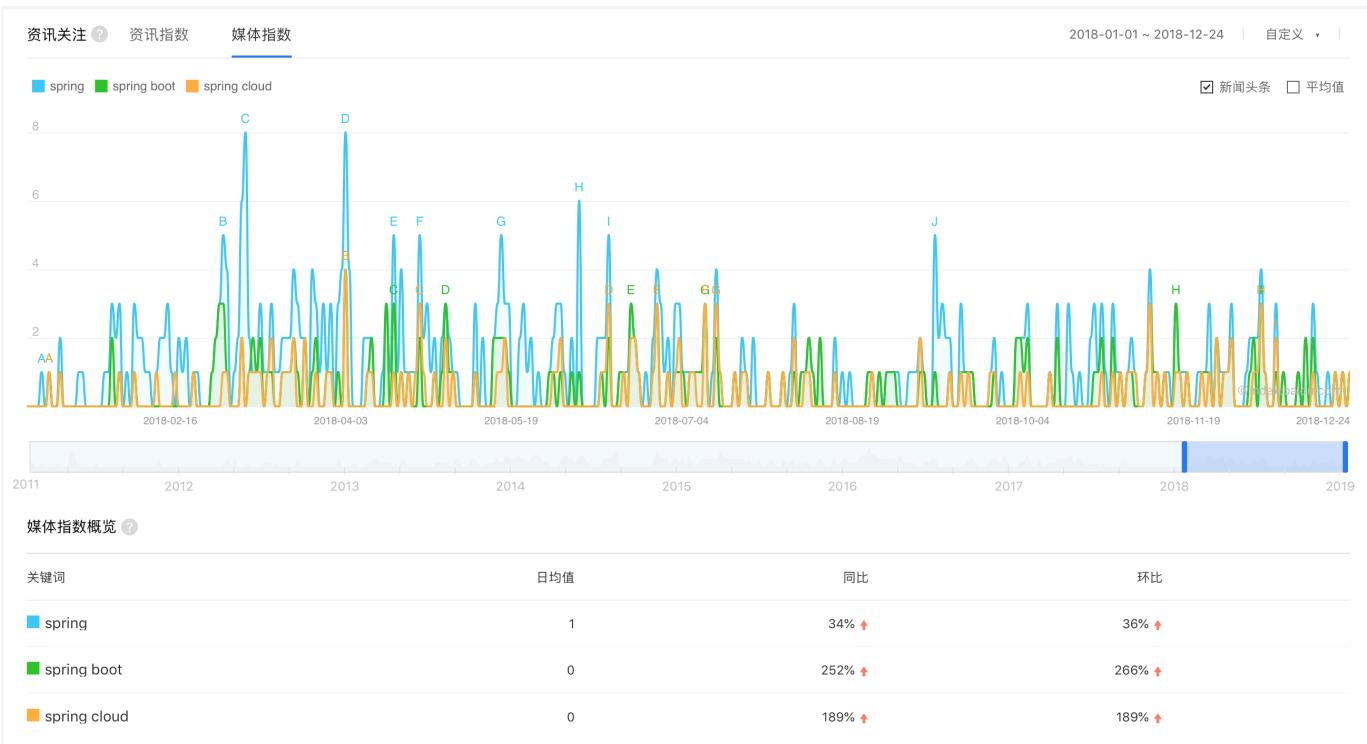


搜索指数

Spring in 2018



# Spring 年终报告



## 版本使用调查

## 结果分析：

1. Spring Cloud 总共有 49 票，目前使用最多的版本是 Finchley，其次是 Edgware。

2. Spring Cloud Greenwich 目前已经发布 RC2，正式版本将于明年1月中旬发布。Finchley 目前需要 Spring Boot 2.0 配合使用。

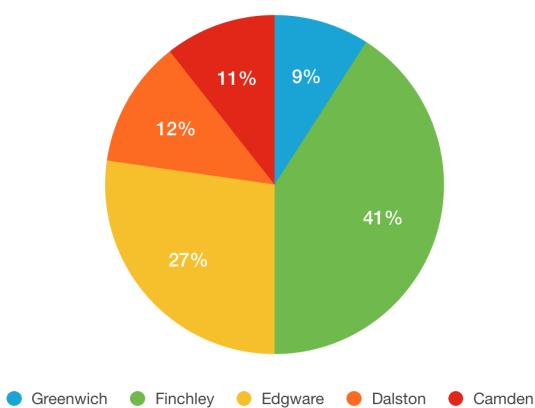
Spring Cloud 使用版本调查结果

版本	投票数
Greenwich	6
Finchley	27
Edgware	18
Dalston	8
Camden	7

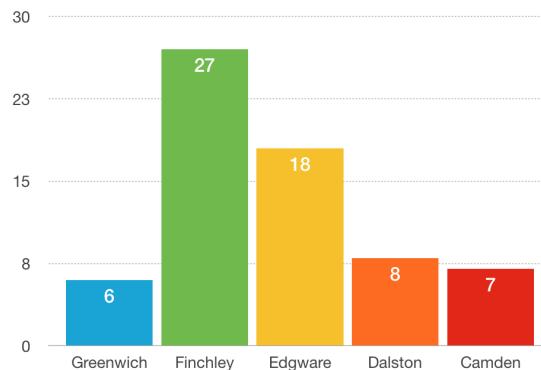
## 结果分析：

1. Spring Cloud 总共有 51 票，目前使用最多的版本是 Finchley，其次是 Edgware。  
2. Spring Cloud Greenwich 目前已经发布 RC2，正式版本将于明年1月中旬发布。  
Finchley 目前需要 Spring Boot 2.0 配合使用。

Spring Cloud 使用版本



Spring Cloud 使用版本



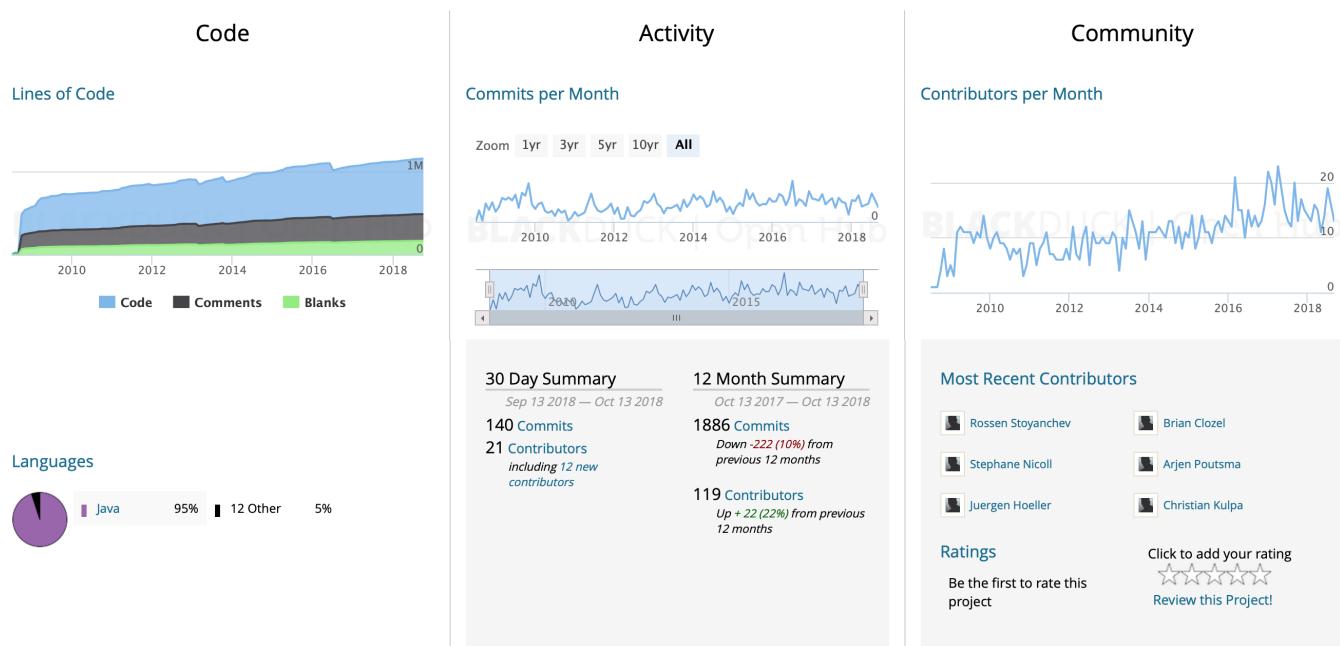
## Spring Framework 项目

## 主要版本发布时间

版本	发布时间
5.1	2018-09-21 07:26:25
5.0	2017-09-28 11:28:23
4.3	2016-06-10 08:59:45
4.2	2015-07-31 09:03:17
4.1	2014-09-04 11:43:07
4.0	2013-12-12 07:15:22
3.0	2009-12-16 16:56:59
2.5	2007-11-19 21:45:36

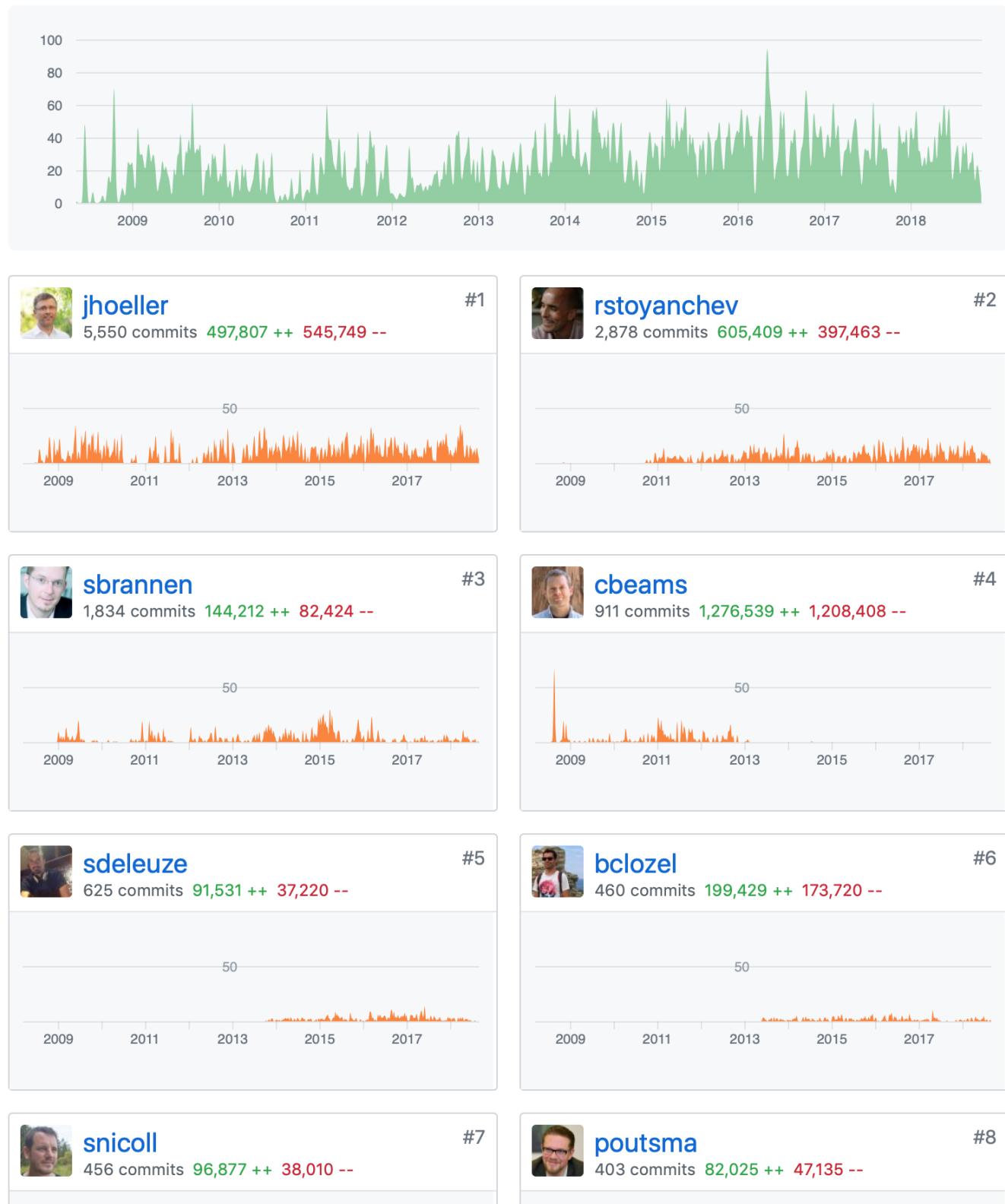
版本	发布时间
2.0	2006-10-03 15:06:33
1.0	2004-03-24 23:21:36

## 项目活跃度



## 项目贡献者

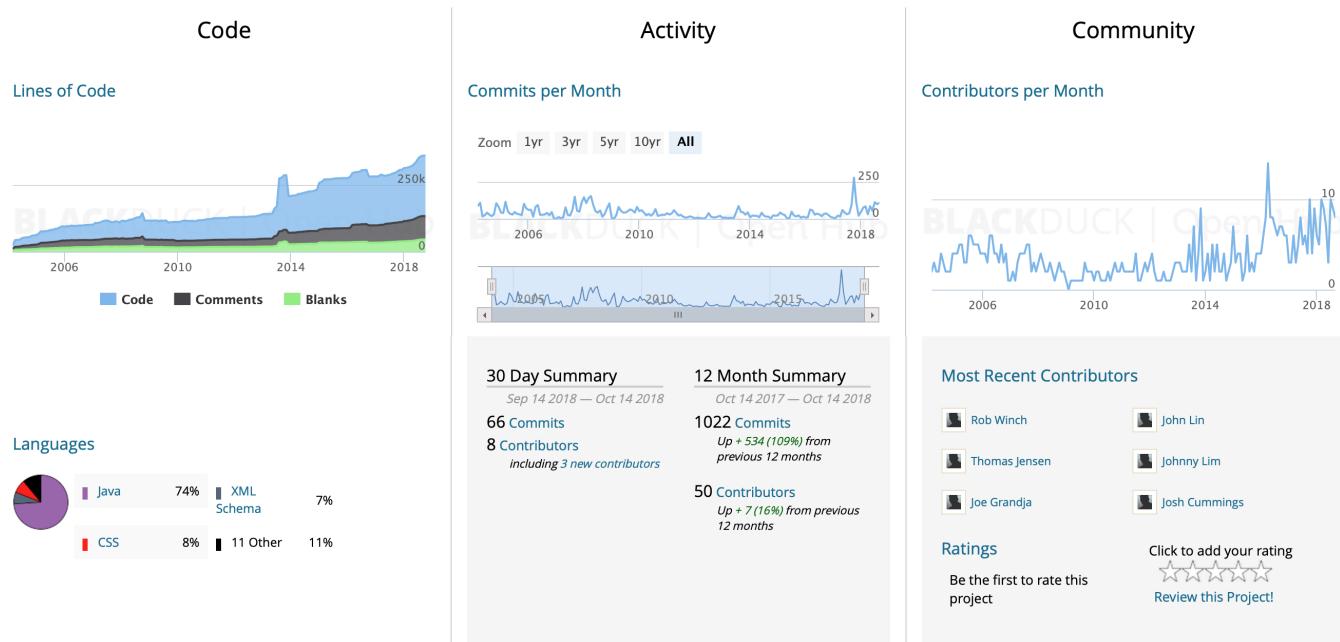
## Spring in 2018



## Spring Security 项目

项目活跃度

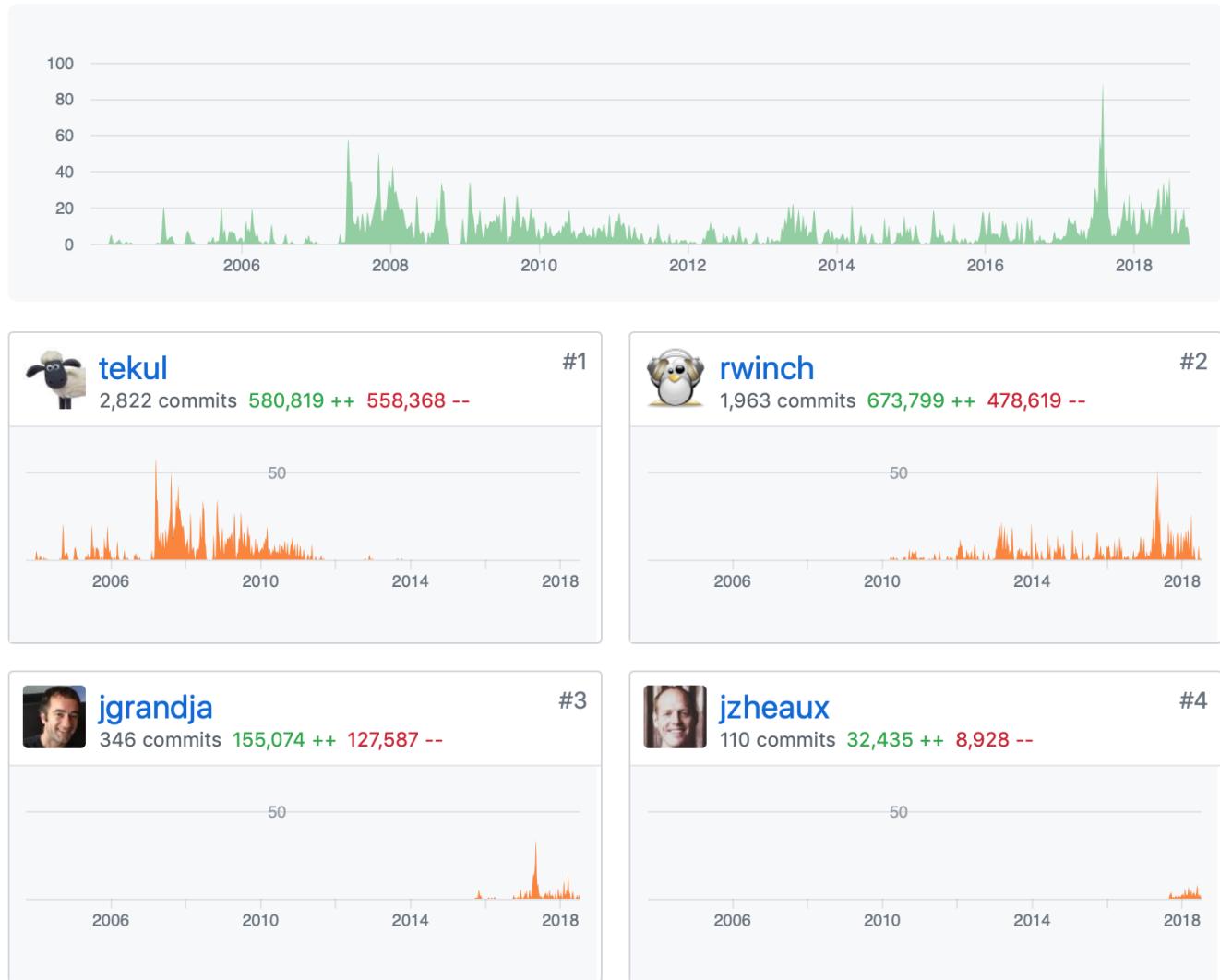
# Spring 年终报告



## 项目贡献者

## Spring in 2018

Contributions to master, excluding merge commits



## Spring Data (JPA) 项目

项目活跃度

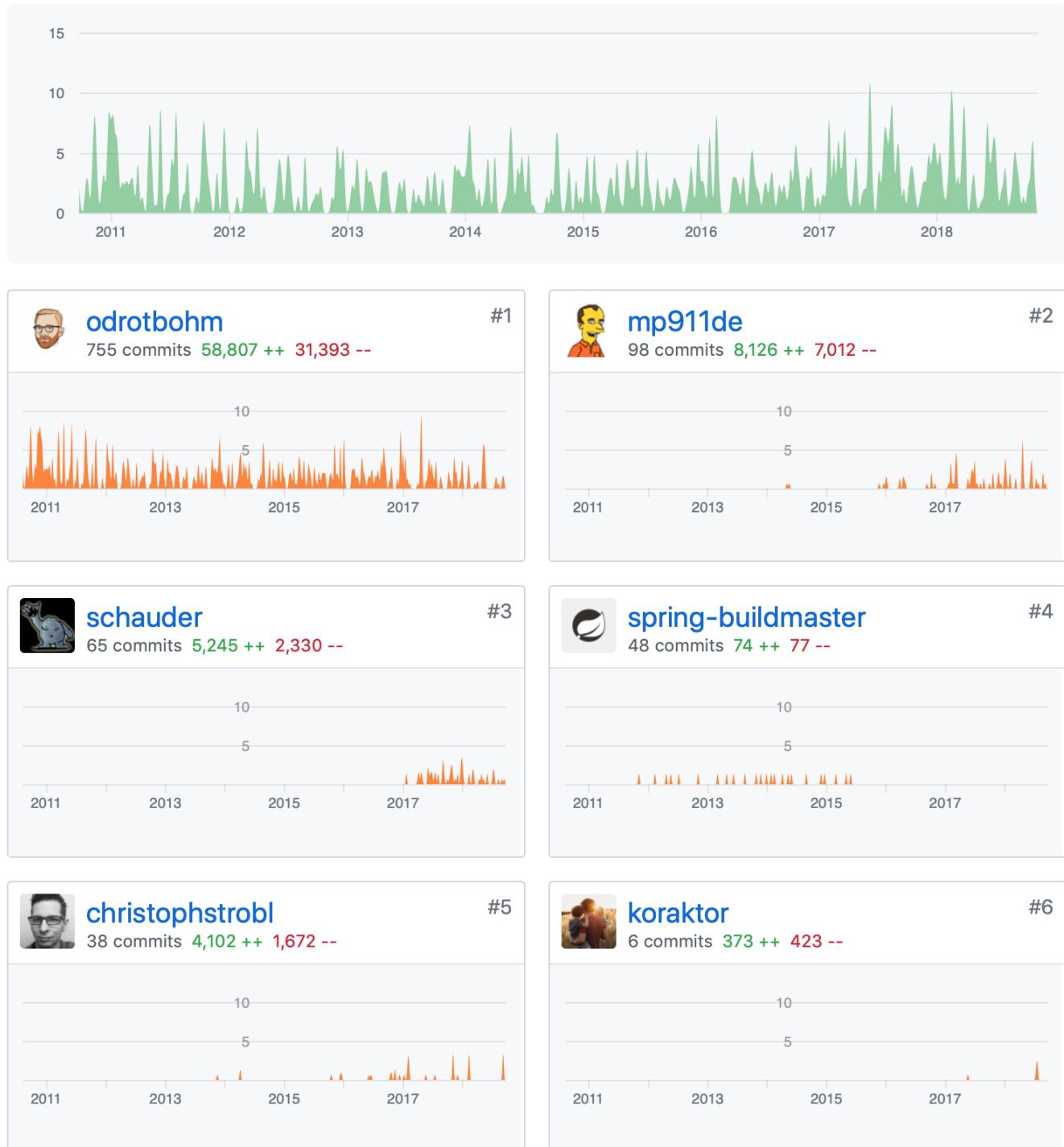
# Spring 年终报告



## 项目贡献者

## Spring in 2018

Contributions to master, excluding merge commits



## Spring Integration 项目

项目活跃度

# Spring 年终报告



## 项目贡献者

## Spring in 2018

Contributions to master, excluding merge commits



## 新年展望

2019 年也是非常重要的一年，又会发生哪些事情，哪些是我们期待的。

语言上，Java 11 开始在生产使用，12、13 将发布；Scala 2.13 将会发布，Groovy 3.0 将会发布，Kotlin 1.4 or 2.0？Go 将会发布 2.0。Spring 是否重启对 .NET 的支持？发布 Spring.NET Framework 3.0？

框架上，微服务框架，多语言，多平台，云和容器支持；Serverless、FaaS。除了 Spring Boot & Cloud之外，其他厂商对框架的投入，比如 Dubbo、Micronaut，都是非常期待 Spring Cloud 继续加强和云服务提供商（Google、AWS、Azure、阿里云、华为云、腾讯云等）的深度合作，更多服务集成到框架中。

工具上，IDE 最强之争，战火重新点燃？JetBrains（IntelliJ IDEA、WebStorm 等）vs Eclipse vs VS Code vs NetBeans，目前 VS Code 在 2018 年增长很快，目前着重还是在前端框架和工具的支持上。2019 年会加大 GitHub 整合、云端开发和部署工具等方面的投入。

移动平台上，目前移动市场显然已经不是 Spring 的重点和优势所在，Spring 虽然有 [spring-mobile](#) 和 [spring-android](#)，但是支持很有限，而且很久没有更新了。

大数据和人工智能方面，目前 Google、Facebook、Microsoft、IBM、阿里巴巴、百度、腾讯等也都有开源一些项目，都在大力投入和积极布局自己的生态，人才、产品、服务、开源、工具等方面竞争无处不在。Google 有 TensorFlow，Facebook 有 PyTorch 和 Caffe2，Microsoft 有 [CNTK](#)，IBM 有 [Watson](#) 和已开源的

[SystemML](#)，云服务提供商可以提供大数据和人工智能的一些中间件产品，使得企业使用很方便能够快速进入这个领域。Apache 有一些开源框架，比如 Hadoop、Spark、Kafka、Flink、Storm，还有一些正在孵化的项目，Spring 开源在框架支持上目前已支持的项目有 [spring-kafka](#) 和 [spring-hadoop](#)。但是 Spring for Hadoop 已有 18 个月未更新，当前最新版本是 2.5.0。

边缘计算和 IoT，Spring 目前并不适合 IoT 设备，2019 年会不会得到改善？另外，来自 Grails 团队今年开源的新项目 Micronaut，在这方面就比较有优势，目前已经发布 1.0.2 版本，期待在 2019 年有更多开发者所采用。

Spring 技术社区，在 2019 年又将如何做？

- 同步翻译官方博客文章，引进新技术、报道新资讯；深度参与项目，提交 issue，PR
- 原创系列，技术选型、解决方案、深度解析、开发指南等等
- 社区活动，在多个城市举办技术沙龙，分享技术、学习交流，  
多种形式：同读源码、读书分享、修复项目问题、翻译用户手册；基础培训、专题培训
- 技术书评，分享最新的技术书籍

# Spring 一周回顾

本章所有文章来自经典系列《This Week in Spring》中文版，一年有 365 天，总共 52 周，而这里有其中的一半——26 周。每周一篇，每篇文章主要来源于 Spring 博客文章，或者 Apache 邮件列表。Spring 官方博客每周会发布一些开源项目的发布说明，而 Spring 技术社区的博客也会从中选择一些文章来翻译，也会关注其他开源项目，比如 Grails、JHipster、Ratpack、Gradle，还有一些前端框架和工具。

## 2018年12月28日

大家好，Spring 粉丝们，感谢一如既往的等待，这将是 2018 年最后一期的《Spring一周回顾》。

2019 年马上就快要到了，祝大家新年快乐，心想事成，财源滚滚，好运连连！



Spring 技术社区，将于 2018 年 12 月 30 日（也就是本年度最后一个星期天）在深圳举办一场聚会沙龙，以技术茶话会的形式，一群 Spring 技术爱好者聚在一起，度过轻松愉快的一个周末时光。届时，将会发布 Spring 2018 年终总结报告，也会有一些主题分享。

目前已经准备好《Spring 2018 年终报告》，主要内容有：

- 2018 年度大事件回顾，回顾一些大家比较关注的事件和话题；
- 2018 年终报告，选择了几个有代表性的项目，分析了 Spring Framework、Spring Boot、Spring Cloud 等开源项目的项目活跃度、贡献者排行、版本发布记录，以及目前大家关注的一些新特性介绍；
- 2019

新年展望，从开发语言、开发工具、技术框架、云计算、大数据和人工智能等几个方面来简要说明了 Spring 项目的一些情况，此外还介绍了 Spring 技术社区的一些新年计划。



本周我为大家带来了如下内容：

- Spring Session for Apache Geode/Pivotal GemFire 2.1.2.RELEASE 发布
- Apache Groovy 2.5.5 已发布
- Apache Flink 1.5.6 已发布
- Apache Log4j Kotlin API 1.0.0 已发布
- Apache NetBeans 10.0 已发布
- Node v11.6.0、v10.15.0、v8.15.0、v6.16.0 已发布
- Ruby 2.6.0 已发布
- Python 3.7.2 已发布、3.6.8 已发布

# 2018年12月21日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

距离 2018 年结束，还剩下最后的一周时间，是时候来一场技术的总结来作为对过去的告别了。

Spring 技术社区，将于 2018 年 12 月 30 日（也就是本年度最后一个星期天）在深圳举办一场聚会沙龙，以技术茶话会的形式，一群 Spring 技术爱好者聚在一起，度过轻松愉快的一个周末时光。届时，将会发布 Spring 2018 年终总结报告，也会有一些主题分享。报名参与方式，下图直接扫描进入【活动行】小程序报名即可，由于场地有限，需审核通过之后才会收到确认邮件。谢谢支持！



本周我为大家带来了如下内容：

- Spring Cloud Greenwich.RC2 发布，正式版计划在2019年1月中旬发布。
- Spring Cloud for Alibaba 0.2.1 发布，增加两个新模块
- Spring Tools 4.1.0 发布
- Spring Tool Suite 3.9.7 发布
- Spring Cloud Open Service Broker 3.0.0.M3 发布
- Spring Cloud Open Service Broker 2.1.0 发布
- Spring Cloud Data Flow and Skipper 2.0 M1 发布
- Spring Cloud Task 2.1.0.M2 发布
- Ratpack 1.6.0 发布
- Micronaut 1.0.2 发布
- JHipster 5.7.2 发布
- Vert.x 3.6.2 发布
- Apache HttpComponents Client 5.0 beta3 已发布
- Apache Knox 1.2.0 已发布

- Apache ServiceComb Saga Actuator 0.3.0 已发布
- Apache Tomcat 9.0.14 已发布
- Apache Tomcat 8.5.37 已发布
- Apache Jackrabbit Oak 1.8.10 已发布
- Apache Tika 1.20 已发布
- Apache Flink 1.7.1 已发布
- Apache Flink 1.6.3 已发布
- Apache Qpid JMS 0.40.0 已发布
- Hibernate OGM 5.4.1.Final 已发布
- Hibernate Search 5.11.0.Final 已发布, 升级到 Hibernate ORM 5.4.0.Final, 支持 JDK 11
- Gradle 5.1.0-RC3 已发布
- Activiti 7.0.97 已发布
- Keycloak 4.8.1.Final 已发布
- Jackson 2.9.8 已发布
- Kibana 6.5.4、Elasticsearch 6.5.4、Logstash 6.5.4 已发布
- Kubernetes 1.12.4、1.11.6、1.10.12 已发布
- Kong 1.0 GA 已发布
- Eclipse IDE 4.10 已发布, 完全支持 Java<sup>TM</sup> 11 和 Java EE<sup>TM</sup> 8
- Visual Studio Code 1.30.1 已发布
- Atom v1.33.1 已发布
- Angular 7.1.4 已发布
- Node v11.5.0 已发布
- Webpack v4.28.2 已发布
- Clojure 1.10.0 已发布
- Kotlin 1.3.20 EAP 2 已发布
- Scala 2.12.8 已发布

# 2018年12月14日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

本周我为大家带来了如下内容：

- Spring Cloud Greenwich.RC1 发布，正式版计划在2019年1月中旬发布。
- Spring CredHub 2.0.0.RC1 发布
- Spring Data R2DBC 1.0 M1 发布
- Spring Data Moore M1 发布
- Spring REST Docs 2.0.3.RELEASE 发布
- Spring REST Docs 1.2.6.RELEASE 发布
- Apache Gobblin (incubating) 0.14.0 已发布
- Apache Calcite Avatica 1.13.0 已发布
- Apache Griffin™ 成为顶级项目 (TLP)
- Apache Jackrabbit 2.19.0 已发布
- Apache Geode 1.8.0 已发布
- Apache Groovy 2.4.16 已发布
- Apache Beam 2.9.0 已发布
- Apache Jackrabbit Oak 1.9.13 已发布
- Apache Lucene 7.6.0 已发布
- Hibernate ORM 5.4.0.Final 已发布，改进实体图，支持 Java 11，升级 JAXB
- Gradle 5.1.0-RC1 已发布，升级 Kotlin DSL 1.1
- Activiti 7.0.89 已发布
- Kibana 6.5.3、Elasticsearch 6.5.3、Logstash 6.5.3 已发布
- Istio 1.0.5 已发布
- Linkerd 2.1 已发布
- Jedis 2.9.1、2.10.0 与 3.0.0 已发布
- Alibaba Nacos 0.7 已发布
- JHipster v5.7.1 已发布
- gRPC 1.17.2 发布，如果你需要在 Spring Boot 项目中集成 gRPC，可以参考这个 [开源的Spring Boot starter](#)
- Visual Studio Code 1.30 已发布
- Angular 7.1.3 已发布
- Node v10.14.2 (LTS) 已发布
- Git 2.20.1 已发布

# 2018年12月7日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

时间总是过得很快，这已经到了2018年第49周，或许还有太多的任务没有完成又或者没有做的更好，但是我们必须停下来开始思考，并且回顾这过去的一年里发生了哪些有意义的事情，重要的不是我们要留住过去（当然我们也无法留住），我们需要从过去的时间中寻找到对自己的肯定和以及对未来的信心，明天才是最重要的，希望都在明天，这才是最令人兴奋的，接下来我们需要足够的勇气并做好迎接它的准备。所以在接下来的几周时间，我将会为 Spring 技术社区的所有开发者们奉上 Spring 2018 年终报告，这对于我来说也的确是个挑战，我希望社区能够为大家带来更多新鲜的东西，尽管还有很多想做的没有做到或者没有做得更好，但是“门已经打开”，那就敞开怀抱准备迎接了。

本周我为大家带来了如下内容：

- [Spring IO Platform Cairo-SR6 发布](#)
- [Spring Tools 4.0.2 发布](#)
- [Grails 3.3.9 发布](#)
- [Apache ServiceComb Java-Chassis 1.1.0 已发布](#)
- [Apache PDFBox 2.0.13 已发布](#)
- [Apache HBase 2.0.3 已发布](#)
- [Apache POI 4.0.1 已发布](#)
- [Apache UIMA Java SDKs 2.10.3、3.0.1 已发布](#)
- [Apache Bahir 2.3.2 已发布](#)
- [Apache Kylin 2.5.2 已发布](#)
- [Apache Jackrabbit 2.18.0 已发布](#)
- [Apache HttpComponents Core 5.0 beta6 已发布](#)
- [Apache CouchDB 2.3.0 已发布](#)
- [Apache Impala 3.1.0 已发布](#)
- [Apache BookKeeper 4.7.3 已发布](#)
- [Apache Ignite 2.7.0 已发布](#)
- [Hibernate ORM 6.0.0.Alpha1 已发布](#)
- [Hibernate Search 5.11.0.CR1 已发布](#)
- [Gradle 4.10.3 已发布](#)
- [Activiti 7.0.87 已发布](#)
- [Angular 7.1.2 已发布](#)
- [Node v11.4.0 已发布](#)
- [Webpack v4.27.1 已发布](#)
- [Kibana 6.5.2、Elasticsearch 6.5.2、Logstash 6.5.2 已发布](#)
- [Istio 1.0.5 已发布](#)
- [Linkerd 2.1 已发布](#)
- [Alibaba Nacos 0.6 已发布](#)
- [SOFABoot v3.1.0 已发布](#)

## Spring 一周回顾

- gRPC 1.17.0 发布，如果你需要在Spring Boot项目中集成gRPC，可以参考这个[开源的Spring Boot starter](#)
- GraalVM Community Edition 1.0 RC10 已发布

# 2018年11月30日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

本周我为大家带来了如下内容：

- Spring Session Bean-SR1 和 Apple-SR7 发布
- Spring Boot 2.1.1 发布
- Spring Boot 2.0.7 发布
- Spring Boot 1.5.18 发布
- Spring IO Platform Brussels-SR15 发布
- Spring Security 5.1.2、5.0.10和4.2.10 同时发布
- Spring Data Lovelace SR3、Kay SR12和Ingalls SR17 同时发布
- Spring Framework 5.1.3、5.0.11和4.3.21 同时发布，这些版本都是一些问题修复。
- Apache Flink 1.7.0 已发布
- Apache Bigtop 1.3.0 已发布
- Apache ServiceComb Saga 0.2.1和Apache ServiceComb Service-Center 1.1.0 已发布
- Apache BookKeeper 4.8.1 已发布
- Apache Jackrabbit Oak 1.9.12 已发布
- Apache Jackrabbit 2.17.7 已发布
- Apache Wicket 7.11.0 已发布
- Hibernate ORM 5.4.0.CR2 已发布
- Hibernate ORM 5.1.17.Final 已发布
- Gradle 5.0.0 正式版发布，Kotlin DSL 1.0 可以正式使用了
- Activiti 7.0.83 已发布
- Angular 7.1.1 已发布
- Node v10.14.1 已发布
- Webpack v4.26.1 已发布
- Atom v1.33.0 已发布
- Serverless 1.34.1 已发布
- Ratpack 1.6.0 RC2 已发布
- R2DBC 1.0 M6 已发布

# 2018年11月23日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

由 Spring 技术社区发起的 Spring 开发者年终聚会沙龙将在下个月底举办，欢迎报名参加或提交演讲主题。

目前已确定演讲主题有两个，也欢迎大家提交内容：

- 《This Year in Spring》，带大家一起回顾2018一年中 Spring 开源和技术社区发生的大事件，并且展望2019年新技术趋势和变革！
- 《Spring Boot最佳实践》，重点讲述了 Spring Boot 在企业级开发中的一些良好实践和经验总结，欢迎一起讨论！

本周我为大家带来了如下内容：

- Spring Cloud Greenwich.M3 发布，RC1 在12月6号，最终正式版RELEASE 将在 12月20号发布。
- Spring Cloud Stream Fishtown.RC2 /2.1.0.RC2 发布
- Spring Cloud Function 2.0.0.RC2 发布
- Apache Camel 2.23.0 已发布
- Apache OpenOffice 4.1.6 已发布
- Apache Kafka 2.1.0 已发布，支持Java 11 以及其他一些新特性
- Apache Jackrabbit Oak 1.6.15 已发布
- Hibernate Search 5.10.5.Final 已发布
- Gradle 5.0.0 RC5 已发布
- Kibana 6.5.1、Elasticsearch 6.5.1、Logstash 6.5.1 已发布
- Resilience4j 0.13.2 已发布
- RxJava v2.2.4 已发布
- Redis 5.0.2 已发布
- Activiti 7.0.77 已发布
- Angular 7.1.0 已发布
- Node v8.13.0 已发布
- Webpack v4.26.0 已发布
- Dubbo 2.6.5 已发布
- Serverless 1.33.2 已发布
- Istio 1.0.4 已发布
- IntelliJ IDEA 2018.3 正式版已发布

# 2018年11月16日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

## Spring技术社区 2018

年终聚会，目前已经开始启动活动策划，欢迎在本年度最后一个星期天（2018年12月30日下午）来参加 Spring 开发者的技术沙龙活动，届时有开源重大资讯发布和 Spring 年度总结汇报！如有更多活动建议和主题分享，可以联系我本人！

本周我为大家带来了如下内容：

- Spring Cloud 团队在官方[Twitter上宣布](#)，Greenwich 的发布计划有些许调整，M2 在[本周发布](#)，RC1 在12月6号，最终正式版RELEASE 将在12月20号发布。
- Spring Session 1.3.4 发布
- Spring Vault 2.1.1、2.0.3 和 1.1.3 发布
- JHipster 5.7.0 已发布，强制使用 UTC，JDL 经过改进，支持直接从 JDL 添加对部署应用程序。
- Micronaut 1.0.1 已发布，利用预编译技术为 Spring 和 Grails 开发者带来福音！
- Apache Groovy 2.5.4 已发布
- Apache Struts 开发团队宣布 2.3.x 进入 End-Of-Life (EOL)
- Apache Bahir 2.1.3 和 2.2.2 已发布，提供多个分布式分析平台的扩展，如Spark 和 Flink等。
- Apache James 3.2.0 已发布
- Apache Juneau 7.2.2 已发布
- Apache Fortress 2.0.3 已发布
- Apache Phoenix 4.14.1 已发布
- Apache Jackrabbit Oak 1.9.11 已发布
- Apache Wicket 8.2.0 已发布
- Apache Qpid JMS 0.38.0 已发布
- Apache Tomcat 7.0.92 已发布
- Gradle 5.0.0-RC2 已发布
- Gradle 5.0.0-RC3 已发布
- Spring Boot Admin 2.1.1 已发布
- Kibana 6.5.0、Elasticsearch 6.5.0、Logstash 6.5.0 同时发布
- Alibaba Nacos 0.5.0 已发布
- SOFABoot v2.5.2 已发布
- Activiti 7.0.67 已发布
- Angular 7.0.4 已发布
- React 16.6.3 已发布
- Keycloak 4.6.0.Final 已发布
- Hibernate ORM 5.4.0.CR1 已发布
- RabbitMQ 3.7.9 已发布
- Serverless 1.33.1 已发布

- Node v11.2.0 已发布
- gRPC 1.16.1 发布，如果你需要在Spring Boot项目中集成gRPC，可以参考这个[开源的Spring Boot Starter](#)
- Visual Studio Code 1.29.1 已发布
- JRuby 9.2.4.0 已发布，支持 Ruby 2.5
- Ratpack 1.6.0 RC1 已发布

# 2018年11月9日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

本周我为大家带来了如下内容：

- Spring CredHub 2.0.0.M1 发布，更好的支持Spring应用与CredHub Server集成
- Spring Cloud Task 2.1.0.M1 发布，支持与Spring Boot 2.1.0
- Reactor Bismuth-SR13 和 Californium-SR2 发布，第三方依赖更新
- Spring Security OAuth2 Auto-config 2.0.6 和 2.1.0 发布，分别支持Spring Boot 2.0.6 和 2.1.0 版本
- Apache Spark 2.4.0 已发布，优化深度学习框架集成，提供更灵活的流式接收器
- Apache Syncope 2.0.11 和 2.1.2 已发布
- Apache Jackrabbit Oak 1.8.9 & 1.9.10 已发布
- Apache Kylin 2.5.1 已发布
- Apache Hive 2.3.4 已发布
- Apache Tomcat 8.5.35 & 9.0.13 已发布
- Apache Kafka 2.0.1 已发布
- Apache Libcloud 2.4.0 已发布
- JHipster 5.6.1 已发布，修复重要BUG
- Alibaba Nacos 0.4.0 已发布
- SOFABoot v3.0.0 已发布，兼容 Spring Boot、Spring Cloud
- Activiti 7.0.65 已发布
- Angular 7.0.3 已发布
- React 16.6.1 已发布
- GraalVM 的Community Edition 1.0 RC9 发布
- TensorFlow 1.12.0 已发布，改善 XLA 稳定性和性能
- Pivotal Greenplum 5.13.0 已发布
- Dgraph v1.0.10 已发布
- RSocket：又一个 REST 的挑战者

# 2018年11月2日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

11月3日，Spring 开源项目背后的公司 Pivotal 在北京举办 SpringOne Tour 技术峰会，包括 Pivotal 的 Spring 技术布道师 Josh Long、Mark Heckler、Spencer Gibb、Paul Czarkowski 等在内的一众 Spring 技术大咖来到中国，还有 Pivotal 中国的资深架构师和专家刘凡、李刚一起为广大 Spring 开发者们献上一道 Spring 技术盛宴。现场总共到了将近 500 多位 Spring 技术爱好者，和大咖们一起讨论技术问题、合影留恋，这是一次非常棒的技术之旅。



大会现场总共安排了 **八** 个专题，详细介绍如下：

1. 《响应式 Spring 和 Spring Boot 2.0》，来自 Pivotal 软件开发布道师 **Mark Heckler**，[演示源码](#)
2. 《云原生 Spring》，来自 Spring 技术布道师 **Josh Long**，闻名不如见面，还是一贯的风格，语速极快、富有激情、语言诙谐，没有 PPT，上来直接撸代码。现场演示构建一个基于 Spring Boot、Reactive、Mongo 的 reservation 微服务，引入 Spring Cloud Gateway、Hystrix、Spring Security、Flow Control、Client 检测、Rsocket 等机制。内容太多，可以看 [视频回放](#)慢慢消化。
3. 《Bootiful Testing》，来自 Spring 技术布道师 **Josh Long**，现场演示 Demo，告诉大家如何测试 Spring 应用和服务，[相关代码](#)
4. 《Spring、函数、无服务器如何为我所用？》，来自 Pivotal 资深云平台架构师 **刘凡**，分享了 Knative、riff 和 Serverless 相关技术实践，这是他的 [GitHub 地址](#)
5. 《Spring Cloud Gateway 的架构和体验之旅》，来自 Pivotal 软件开发布道师 **Spencer Gibb**，详细介绍了 Spring Cloud Gateway 2.0 的技术原理和特性，这是他的 [GitHub 地址](#)
6. 《为什么 Cloud Foundry 是运行 Java 微服务的最佳平台？》，来自 Pivotal 高级解决方案架构师 **李刚**，现场分析了企业微服务开发的痛点，介绍了 Pivotal Cloud Foundry

的强大功能，以及如何快速部署 Spring Boot 应用到生产环境

7. 《使用 Spinnaker 在 Kubernetes 上创建开发工作流》，来自 Pivotal 解决方案架构师 **Paul Czarkowski**，Paul 介绍了 k8s 及其核心组件，借助 Spinnaker 的演示，着重介绍了 Spinnaker 可以轻松创建自定义工作流，用于在 k8s 上测试、构建和部署应用。这是他的 [Github 地址](#)
8. 《借助 Spring Cloud Stream 2.0 实现云事件驱动的架构》，来自 Pivotal 的 Spring 技术布道师 **Jakub Pilimon**，很可惜由于签证问题没能来得大会现场，对这个主题感兴趣的朋友可以在他的 [博客](#) 和 [GitHub](#) 了解相关内容

另外，还有来自 Pivotal 中国官方的现场报道，内有大会现场高清图片和视频地址：

- Spring 官方技术盛会：SpringJosh Long等大神Show代码聊云原生还说了啥
- Josh Long的云原生 Spring 现场视频回放

接下来，本周的《Spring一周回顾》我为大家带来了如下内容：

- Spring Tools 4.0.1 发布
- Spring Session Bean GA 发布，支持Java 11和Spring Boot 2.1
- Spring Cloud for Alibaba 0.2.0 发布，集成支持阿里巴巴开源项目Nacos 0.3.0和Sentinel 1.3.0，以及阿里云的EDAS、ACM、NAS、OSS等中间件产品。
- Spring Boot 2.1.0 正式发布，支持Java 11，升级第三方依赖到最新版本、性能提升、Metrics升级
- Spring Batch 4.1 正式发布，带来了大量新特性
- Spring Integration 5.1 正式发布
- Spring for Apache Kafka 2.2 发布，需要 **kafka-clients** 2.0.0，同时提供诸多改进和增强。此外，Spring Integration for Apache Kafka (spring-integration-kafka) 3.1.0.RELEASE 也已发布，它基于Spring for Apache Kafka 2.2和Spring Integration 5.1构建
- Spring Data Lovelace SR2 发布，基于Spring Framework 5.1.2，共修复了30个问题。
- Spring Cloud Function 2.0.0.RC1 发布，支持Kotlin，基于Spring Boot 2.1构建
- Spring Framework 5.1.2 发布，解决了30个问题单。
- Apache Nifi 1.8.0 已发布
- Apache Flink 1.6.2和1.5.5 已发布
- Apache HBase 2.1.1 已发布
- Apache XMLBeans 3.0.2 已发布
- HttpComponents Client 5.0 beta2 已发布
- Apache BVal 2.0.0 已发布，实现Java Bean Validation 规范 2.0，APL协议
- Apache Juneau 7.2.1 已发布
- Apache Jackrabbit 2.12.10 已发布
- Apache Hive 3.1.1 已发布
- Apache Subversion 1.11.0 已发布
- Apache Maven 3.6.0 已发布
- Gradle 5.0.0-RC1 已发布，支持JDK 11，Kotlin DSL 1.0可用生产项目
- Hibernate OGM 5.4.0.Final 已发布
- JHipster 5.6.0 已发布，升级到Node 10、TypeScript 3、Angular 7、Spring Boot 2.0.6

- Apollo v1.1.2 已发布
- Alibaba Nacos 0.3.0 已发布
- Alibaba Sentinel 1.3.0-GA 已发布，如何将Sentinel 控制台应用于生产环境
- Google Guice 4.2.2 已发布，支持Java 11
- Netty 4.1.31.Final 已发布
- Kotlin 1.3 已发布
- Activiti 7.0.58 已发布
- TypeScript 3.1.6 已发布
- Node v11.1.0 已发布
- Angular 7.0.2 已发布
- Atom 1.32.1 发布

# 2018年10月26日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

在 Oracle Code 2018 大会上，首先在这里要恭喜 Graeme Rocher 与 Doug Cutting (Apache Hadoop联合创始人)、Neha Narkhede (Apache Kafka联合创始人&CTO)、Charles Nutter (JRuby项目领导人之一)、Guido van Rossum (Python发明者) 一起获得了 Oracle 开创性大奖 (Groundbreaker award)。Oracle 创新者奖颁发给为开源软件生态系统做出了重大贡献的技术专家，Graeme 因在共同创建和开发 Grails Web 应用程序框架方面的开创性工作而受到表彰。

他也是新一代微服务框架 Micronaut 的主要开发者之一，Micronaut 于10月23日已经 正式发布！如果您目前还不了解这个框架，可以查看之前的版本发布说明和框架介绍：

- Micronaut 正式开源，构建微服务应用新选择！
- Micronaut 1.0 RC1 发布，了解其独特的四个关键特性
- Micronaut 1.0 RC2 发布，详细介绍了其实现技术及原理
- Micronaut 1.0 RC3 和 Micronaut Test 1.0 RC1 发布
- Graeme Rocher 访谈：介绍 Micronaut

还有来自InfoQ的翻译文章：

- 全栈JVM框架Micronaut通向1.0版本之路
- Micronaut教程：如何使用基于JVM的框架构建微服务

接下来，本周的《Spring一周回顾》我为大家带来了如下内容：

- Micronaut 1.0 正式发布，Grails团队出品，值得信赖，欢迎使用。
- Spring Session for Apache Geode/Pivotal GemFire 2.0.6.RELEASE 和 2.1.0.RELEASE 同时发布，允许用户根据会话到期的方式和时间自定义规则，也支持固定到期超时时间设置。
- Spring Cloud Data Flow 1.7 正式发布，改进UI、流应用程序DSL、审计跟踪、流和任务校验、支持在Kubernetes上调度任务等
- Spring Cloud Finchley.SR2 发布，多个模块更新。
- SOFABoot v2.5.1 已发布
- Spring Boot Admin 2.0.4 已发布
- gRPC 1.16.0 发布，如果你需要在Spring Boot项目中集成gRPC，可以参考这个开源的Spring Boot starter
- Apache HttpComponents Core 5.0 beta5 已发布，此BETA版本增加了对Reactive Streams API的支持[<http://www.reactive-streams.org/>]，并修复了兼容性问题Java 11新的TLS引擎，以及自此以来发现的上一版本的一些缺陷。
- Apache ServiceComb 毕业成为TLP，来自InfoQ的新闻：微服务开源项目ServiceComb 毕业成为Apache顶级项目
- Apache Sling 11 已发布，升级到OSGi R7和Oak 1.8.8
- Apache Impala 3.0.1 已发布
- Apache Pulsar 2.2.0 已发布
- Apache Kudu release 1.8.0 已发布
- Kubernetes 1.12.2、1.11.4 已发布

- Activiti 7.0.54 已发布
- OrientDB v3.0.10 已发布
- MySQL Community Server 8.0.13 已发布
- Node v11.0.0 已发布，升级V8 7.0
- Angular 7.0.1 已发布
- Atom 1.32.0 发布

# 2018年10月19日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

中国开源年会 COSCon'18 于本周末在深圳南山举办，两天的时间，来自国内外众多开源爱好者参与了大会，其中来自有 Apache 的VP Justin Mclean、GitHub 未来 CEO Nat Friedman 等国内外技术大咖们带来了主题分享。此外还有微软、华为、阿里、腾讯等相关产品和开源的技术分享，作为本次大会的主办方开源社也在大会上正式发布了 2018 年中国开源年度报告。

- 2018年中国开源年度报告正式发布 | COSCon'18 特辑
- GitHub发布史上最大更新，年度报告出炉！
- GitHub CEO 参加COSCon ‘18
- 腾讯三大运维开源项目齐聚“OSCAR开源先锋日”

接下来，本周的《Spring一周回顾》我为大家带来了如下内容：

- Spring Cloud Data Flow 1.7 RC1 发布，基于1.7 M1中引入的核心功能，并进行了一些改进。
- Spring IO Platform Cairo-SR5 发布，升级多个项目依赖。
- Spring IO Platform Brussels-SR14 发布，升级多个项目依赖。
- Spring Cloud Edgware.SR5 发布，部分模块更新。
- Spring Security 5.1.1、5.0.9和4.2.9 同时发布，主要是解决部分缺陷、升级第三方依赖以及优化。
- Spring Boot 2.1.0 RC1 发布，正式版预计在月底发布。
- Spring Boot 2.0.6  
发布，共解决了97项问题修复、改进和依赖项更新，以及2个安全问题，建议使用2.0.x的用户升级到此版本。
- Spring Boot 1.5.17 发布，共解决了19项问题修复、改进和依赖项更新。
- Spring Framework 5.1.1、5.0.10和4.3.20 同时发布，修复一些问题。
- Spring Data Lovelace SR1、Kay SR11和Ingalls SR16  
同时发布，主要包含错误修复和一些依赖性升级，总共修复了70多个问题。
- Micronaut 1.0 RC3和Micronaut Test 1.0 RC1 发布，GA版本将在10月23日发布。
- Apache Struts 2.3.36和2.5.18 同时发布
- Apache Beam 2.7.0 已发布
- JHipster 5.5.0 已发布，关闭了84个问题和PR。
- HashiCorp Vagrant 2.2.0 已发布，提供了新工具：Vagrant Cloud CLI
- GraalVM 的Community Edition 1.0 RC8 发布
- Redis 5.0 已发布，提供新的流数据类型，新的模块API：Timers, Cluster and Dictionary等
- Visual Studio Code 1.28.2 已发布
- TypeScript 3.1.3 已发布
- Angular 7.0.0 正式发布

# 2018年10月12日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

在Java领域，开源的框架一直都非常多，从WebWork、Struts、Tapestry、Grails、Dropwizard、Ratpack、Spark，包括 Spring Boot，这些都是非常优秀的开源框架。Micronaut 被大多数 Spring & Grails 的开发者寄予厚望，今年5月23日来自 OCI 的 Graeme Rocher 宣布了 Micronaut 正式开源之后，开发团队一直非常努力投入开发之中，在接连发布了 M1、M2、M3、M4 以及 RC1、RC2 等版本之后，终于离最终正式版越来越近了，官方计划在10月16号发布第三个候选版本，不出意外将在10月23日迎来最终的 GA 版发布。欢迎关注本网站，后续会继续报道最新资讯和推出相应学习教程。

接下来，本周的《Spring一周回顾》我为大家带来了如下内容：

- 今年的5月，OCI宣布 Micronaut 在 GitHub 开源了，我为你翻译了其中的 RC1 的发布说明，介绍了其独特的四个特性；RC2 的发布说明，详细介绍其实现技术及原理揭秘。
- Thymeleaf 3.0.10.RELEASE 已发布，已在JDK 11测试通过；Thymeleaf Spring Security 3.0.3 也同时发布，支持Spring Security 5
- Infinispan 9.4.0.Final 已发布
- HashiCorp Consul 1.3 已发布，支持Envoy作为Connect的代理，并在Kubernetes中启用自动Sidebar注入以实现安全的pod通信
- JHipster 5.4.2 已发布，升级依赖JHipster Core v3.4.0，支持Azure Pipelines
- Redis客户端Lettuce 5.1.1 已发布
- Apache Arrow 0.11.0 已发布
- Apache Groovy 2.5.3 已发布，修复20+BUG，多个依赖项升级
- Apache OFBiz® 16.11.05 已发布
- Apache Subversion 1.10.3 已发布
- Apache Tika 1.19.1 已发布
- Apache Velocity Tools 3.0 已发布，依赖Velocity Engine 2.0
- Visual Studio Code 1.28.1 已发布
- TypeScript 3.1.2 已发布
- Node v10.12.0 已发布
- Angular 6.1.10 和 7.0.0 rc1 同时发布
- Ansible 2.7.0 已发布
- Apollo v1.1.0 已发布
- Grafana 5.3 已发布
- SOFABoot v2.5.0 已发布

# 2018年10月5日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

这个十一长期已经结束，相信大家一定玩的开心，那么接下来将会有一系列的视频内容等着我们去学习和消化，在9月24-27日为期四天的SpringOne Platform 2018大会上，来自业界的技术大咖们分享了最新的技术资讯、最佳实践和行业案例介绍。官方也已经在这几天在YouTube上放出了全部视频：SpringOne Platform 2018视频清单，下面是我为大家整理的其中一些主题演讲及示例代码：

- Bootiful Testing, [示例代码](#), Josh Long
- The New Kid on the Block: Spring Data JDBC, [示例代码](#), Jens Schauder
- Scaling Spring Boot Applications in Real-Time, [示例代码](#), Luke Shannon和John Blum
- Building Cloud-Native Data-Intensive Applications with Spring, [示例代码](#): eda
- Michael Minella分享的Keynote视频, [示例代码](#)和胶片
- Serverless with Spring Cloud, [示例代码](#), Thomas Risberg
- Demystifying SAML Using Spring Security, [代码](#)和PPT, Filip Hanik和Sree Tummidi
- Spinnaker and the Distributed Monorepo: [演示代码](#), Jon Schneider
- Beyond Windows - Hacking Cloud Apps on Linux and .NET for the Busy Java Developer, [示例代码](#), Brian Benz和Maxime Rouiller
- MongoDB + CredHub = Secure By Default Data Services on PCF, [示例代码](#), Peter Blum和Diana Esteves
- Consumer Driven Contract Testing with Spring Cloud Contract, [示例文档](#), Eddu Melendez
- Global Event Streams Made Simple with Spring Cloud Stream & Cloud Pub/Sub, [示例代码](#), Artem Bilan
- Netifi Proteus and RSocket的[演示代码](#), Ryland Degnan

与此同时，本周的《Spring一周回顾》我为大家带来了如下内容：

- Spring Data Lovelace 正式发布，有关其新特性的说明看[这里](#)，以及Spring Data JDBC模块的特性介绍。
- Spring Cloud Open Service Broker 3.0.0.M1 发布
- Spring Cloud Open Service Broker 2.1.0.M1 发布
- Spring Vault 2.1 正式发布
- Spring Fu 0.0.2 发布，提供改进的Kofu DSL并引入功能参数的自动装配，这里有[详细介绍](#)
- Apache Geode 1.7.0 已发布
- Apache PDFBox 1.8.16和2.0.12 已发布
- Apache Qpid JMS 0.37.0 已发布
- Apache Subversion 1.11.0-rc2 已发布
- Nginx 1.15.5 已发布
- Julia 1.0.1 已发布
- Angular 7.0.0-rc.0 已发布
- TypeScript 3.1.1 已发布
- Neo4j 3.4.8 已发布

- MariaDB 10.3.10 已发布
- Kibana 6.4.2、Elasticsearch 6.4.2、Logstash 6.4.2 同时发布
- 服务监控系统 Prometheus 2.4.3 已发布
- fastjson 1.2.51 已发布，这又是一个BUG修复安全加固版本
- Gradle 5.0.0-M1 已发布，4.x版本中已弃用的许多内容已被删除，运行需要Java 8+
- JHipster 5.4.0 和 JHipster 5.4.1 相继发布，使用 Jib 来创建Docker镜像，支持MongoDB的丰富文档模型（嵌入和引用），升级依赖JHipster Console v4.0.0、K8s Istio v1.x、Keycloak 4.5.0.Final
- GraalVM 的1.0 RC7 发布，如需进一步了解Graal，可以查看[解密新一代Java JIT编译器Graal](#)
- Java 11正式发布，这里有[新特性解读](#)和[又一篇新特性介绍](#)
- KotlinConf 2018于10月3-5日在线直播，这里可以了解到[Kotlin 1.3 RC的新功能](#)

接下来的相关社区活动：

- 2018中国开源年会  
COSCon'18，将会在2018年10月20、21两天在深圳南山区举办，欢迎一起前往参加这个开源聚会。
- SpringOne Tour 2018  
技术峰会，2018年11月3日在北京京仪大酒店，来自Pivotal美国的Spring明星大咖和国内技术专家组成了豪华阵容，他们将与大家分享并探讨现代应用开发、DevOps、CI/CD、云计算等话题。

# 2018年9月28日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

本周 Java 界最重大事件，莫过于 Oracle 官方在北京时间9月26号宣布 [Java 11 正式发布](#)，这是 Java 大版本周期变化后的第一个长期支持版本，非常值得关注。Spring 5.1 也于上周正式发布，支持 Java 11。而 Spring 4.3.x 确定不会支持 Java 9+，所以如果要使用 Java 11，那么是时候升级到 Spring 5.1 了。Spring Boot 2.1 也将于下个月底发布正式版。

[SpringOne Platform](#)在本周的9月24-27日在 Washington D.C 成功举办，其中来自业界的技术大咖们分享了最新的技术，也有一些来自行业的案例分享。有关大会相关的视频和PPT，稍后会整理放出来，敬请关注。

本周我为大家带来了如下内容：

- [Spring Tools 4 正式发布](#)，这是一次完全的重构，专为现代化 Spring 技术而构建的新工具，包括智能感知、提供运行应用的实时信息、重大性能提升等，除了支持Eclipse，还支持Visual Studio Code和Atom IDE。
- [Spring Boot 2.1.0 M4 发布](#)，构建在 Spring Framework 5.1 GA、Spring Data Lovelace-RELEASE、Spring Security 5.1.0.RELEASE 和Reactor Californium 等最新的版本之上，GA 版本将于10月底发布。
- [Spring Framework 5.1.0 发布](#)，需要JDK 8及以上版本，特别是支持 JDK 11 这个 LTS 版本。优化了启动时间和内存消耗，升级到 Reactor Californium、Hibernate ORM 5.3。
- [Spring Security 5.1.0 发布](#)
- [Spring Batch 4.1.0.RC1 发布](#)
- [Spring Data Lovelace 正式发布](#)，新增Spring Data JDBC模块，这里有篇介绍其特性的文章。
- [Spring Web Services 3.0.4和2.4.3 同时发布](#)，支持Java 11。
- [Spring Boot for Apache Geode和Pivotal GemFire 1.0.0.M3 发布](#)。
- [Spring Session BOM Bean-RC1 发布](#)
- [Apache Lucene 7.5.0和Apache Solr 7.5.0 同时发布](#)
- [Apache Juneau 7.2.0 已发布](#)
- [Apache HBase 1.2.7 已发布](#)
- [TensorFlow 1.11.0 已发布](#)
- [Keycloak 4.5.0.Final 已发布](#)
- [Netty 4.1.30 发布](#)，支持Java 8以上版本，包括Java 11
- [Kubernetes 1.12.0 已发布](#)
- [Nacos 0.2.1 已发布](#)，这里有个支持Spring Boot的项目：[Nacos for Spring Boot](#)
- [Atom 1.31.0和1.31.1 相继发布](#)
- [Micronaut 1.0 RC1 发布](#)，这是一次重要的里程碑，说明所有特性已经开发完成，其新特性包括支持构建GraalVM原生镜像，编译时校验和编译时生成Swagger文档，借助最前沿的AOT（预先编译）技术，相信它未来在微服务、IoT、无服务等场景中将会发挥重大作用。
- [Java 11正式发布](#)，这里有[新特性解读](#)，[下载地址](#)

# 2018年9月21日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

中秋节即将到来，首先祝大家节日快乐！海上升明月，天涯共此时。无论我们在哪里，家的方向就是前进的方向，家人的支持和陪伴就是前行的动力，我在此祝所有Spring开发者们，阖家欢乐，幸福安康。但愿人长久，千里共婵娟！

本周我为大家带来了如下内容：

- Spring Framework 5.1.0 发布，需要JDK 8及以上版本，特别是支持JDK 11这个LTS版本。优化了启动时间和内存消耗，升级到Reactor Californium、Hibernate ORM 5.3。
- Spring Tool Suite 3.9.6 发布，升级到Eclipse 4.9版本。
- Spring Cloud Function 2.0.0.M2 发布，支持Kotlin
- Spring Cloud Data Flow 1.7 M1  
发布，其中不少亮点：改进UI，流应用程序DSL，审计跟踪，并发任务启动限制，流和任务校验，强制升级流应用等。
- Apache Flink 1.6.1和Apache Flink 1.5.4，同时更新发布
- Apache Tomcat 7.0.91 已发布
- Apache Tika 1.19 已发布，需要JDK 8，解决安全漏洞
- Apache Kylin 2.5.0 已发布，支持Hadoop 3.0和MySQL
- Apache Pulsar 2.1.1 已发布
- Apache JMeter 5.0 已发布，大量新特性及优化
- RabbitMQ 3.7.8 已发布，升级到 Erlang 21.0
- Jackson 2.9.7 已发布
- Kotlin 1.3 RC 已发布
- Eclipse 4.9 已发布
- IntelliJ IDEA 2018.2.4 已发布
- Kibana 6.4.1、Elasticsearch 6.4.1、Logstash 6.4.1 同时发布
- Jenkins 2.138.1 和 2.141 已发布
- Gradle 4.10.2 已发布，解决 Scala 项目依赖性问题
- Kubernetes 1.12.0-rc1、1.10.8 已发布
- Linkerd 2.0 已发布
- Kong 1.0.0.rc1 已发布
- Serverless 1.32.0 已发布
- Tsuru 1.6.0 已发布
- JHipster v5.3.4 已发布
- Nacos 0.2.1-RC1 已发布

接下来的相关社区活动：

- SpringOne Platform将9月24-27日在Washington D.C举办，届时将有众多业界大咖们作为演讲嘉宾，比如熟知的Rod Johnson、龙之春、Matt Raible

、 Phillip Webb

- ASF的官方全球会议ApacheCon，将于9月24日-27日在加拿大的蒙特利尔举办，同时也是Apache社区庆祝20周年的一次盛大活动。
- SpringOne Tour 2018  
技术峰会，2018年11月3日在北京京仪大酒店，来自Pivotal美国的Spring明星大咖和国内技术专家组成了豪华阵容，他们将与大家分享并探讨现代应用开发、DevOps、CI/CD、云计算等话题。

# 2018年9月14日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

在台风天里写作，是一次难得的体验。此刻，狂风呼啸，大雨漂泊，在这座城市里每一个角落，无论是风雨中屹立的建筑物，还是随风摇摇晃晃的树木，以及暴雨洗刷的道路，还有行驶在道路上的车辆，匆匆忙忙正在赶回家的行人，无不都在经受着最严峻的考验。

本周我为大家带来了如下内容：

- [Spring Security 5.1.0.RC2 发布](#)，新版本支持大量新内容，包括用于OAuth 2的简化DSL，WebClient扩展，servlet增强，改进的资源服务器支持，改进的RestTemplate支持，对X.509的支持以及使用环境变量来配置LDAP。
- [Spring IO Platform Brussels SR13 发布](#)
- [Spring IO Platform Cairo SR4 发布](#)
- [Spring Boot 2.1.0 M3 发布](#)
- [Spring Boot 2.0.5 发布](#)
- [Spring Boot 1.5.16 发布](#)
- [Spring Security 5.0.8和4.2.8 同时发布](#)，主要是解决部分缺陷、升级第三方依赖以及优化。
- [Spring Data Ingalls SR15 和 Kay SR10 同时发布](#)，修复了一些Bug和升级一些第三方依赖。
- [Spring Vault 2.0.2 发布](#)，解决部分缺陷、升级第三方依赖以及优化。
- [Spring Vault 2.1 RC1 发布](#)，Spring Vault 2.1需要JDK 8或更高版本，并且特别支持JDK 11作为下一个长期支持版本。这个版本完成了15张任务单，以及支持Java 9到11所做的一些改进。
- [Spring Framework 5.1 RC3, 5.0.9 和 4.3.19 同时发布](#)
- JUnit 5.3 已发布，它包括并行测试执行，输出捕获，新的TestInstanceFactory Extension API等等
- [Apache Kylin 2.4.1 已发布](#)
- [Apache Tomcat 8.5.34和9.0.12 已发布](#)
- [Apache Qpid Proton 0.25.0 已发布](#)
- [Apache POI 4.0.0 已发布](#)
- [ActiveMQ 5.15.6 已发布](#)
- [Apache Wicket 8.1.0 已发布](#)
- [SOFABoot v2.4.8 已发布](#)
- [Kotlin 1.2.70 已发布，增量编译速度提高7倍](#)
- [RocksDB 5.15.10 已发布](#)
- [服务监控系统 Prometheus 2.4.0 已发布，包含多项修正和改进](#)
- [Neo4j 3.3.7 已发布](#)
- [Dubbo 2.6.3 正式版已发布](#)，带来了功能增强、新特性、bug 修复、性能优化和Hessian-lite。
- [GRPC 1.15.0 已发布](#)

Apache HBase技术社区第五届MeetUp本周六下午在深圳阿里中心举办，下面放出几个主题的链接：

- [HBase应用与发展之Apache HBase的现状和发展](#)

- HBase应用与发展之HBase应用与高可用实践
- HBase RowKey与索引设计

## 2018年9月7日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

九月正是中国学生开学的季节，在此我真诚为各位奉上 [《Spring 最全书单》](#)，送给爱学习的你们，希望大家学习和工作都有好的收获。

本周我为大家带来了如下内容：

- Spring Framework 5.1 RC3, 5.0.9 和 4.3.19 同时发布，下周将会发布相应的Spring Boot 更新版本
- Spring Batch 4.1.0.M3 发布，主要是添加对JSR-305注解的支持，改善代码可读性，与IDE集成可给开发提供一些接口的有用信息。
- Spring Boot for Apache Geode 和 Pivotal GemFire 1.0.0.M2 发布
- Spring Session for Apache Geode/Pivotal GemFire 2.0.5.RELEASE 和 2.1.0.M1 同时发布
- Apache HBase 2.0.2 发布了，解决了大约100多个問題修复，旨在提高稳定性和可靠性，建议升级。
- Keycloak 4.4.0 发布，升级到WildFly 13，授权服务支持NodeJS集成，提供细粒度的中央授权适配器。
- JHipster 5.3.0和 JHipster 5.3.1 本周相继发布！
- GraalVM 的1.0 RC6 发布，如需进一步了解Graal，可以查看[解密新一代Java JIT编译器Graal](#)
- Node v10.10.0 发布
- Angular 6.1.7 和 7.0.0 beta5 发布
- React 16.5.0 发布
- CNCF接纳Harbor为沙箱项目
- 构建一个运行在Azure虚拟机上的MySQL Spring Boot应用程序

# 2018年8月31日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。

十五年前的今天是 [Groovy](#) 语言诞生的日子，创始人 [James Strachan](#)首次提交代码。Groovy 语言与2007年2月正式发布 [1.0](#)，2012年7月发布最重要版本 [2.0](#)，2015年4月在与 SpringSource 分手之后，加入 Apache 开源基金会后同年11月毕业，截止目前总共有 [160](#) 多次发布，超过 [一亿](#) 次下载量，总共有 [320+](#) 开发者参与贡献代码，其中 [Paul King](#) 提交次数最多，贡献了 3702 次。  
时光飞逝，我是在2004年首次接触 Groovy 语言，之后开始了学习之路，算是国内最早一批接触 Groovy 语言和 Grails 框架的人了，伴随着 Groovy 和 Grails、Spring 的成长和壮大，追随着开源社区的诸多前辈大神，一路走来，这都是一段非常棒的经历和美好回忆。

本周我为您带来了如下内容：

- [Spring Session for Apache Geode/Pivotal GemFire 2.0.5.RELEASE 和 2.1.0.M1 同时发布](#)
- [Spring Security OAuth2 Boot Auto-config 2.0.4 和 2.1.0.M2 同时发布](#)
- [WildFly 14 正式发布](#)，通过Java EE8 认证
- [Akka 2.5.16 发布](#)，修复安全问题
- [Gradle 4.10 正式发布](#)，此次默认启用Java的增量编译功能，此外包括了 Kotlin DSL 也发布1.0 RC1。
- [Go 1.11 发布](#)，初步支持“模块”
- [Kotlin 1.3-M2 发布](#)，带来了新特性
- [MongoDB 4.0.2 发布](#)
- [Angular 6.1.6 发布](#)，修复部分 BUG
- [Atom 1.30 发布](#)，升级到 Electron 2.0.4 及问题修复和更新
- [SOFABoot 2.4.7 发布](#)
- [Cloud Foundry 基金会给出调查报告指出 Java 和 JavaScript 两种语言在企业开发中依然最受欢迎](#)
- [Babel 7.0 正式发布](#)，支持 TypeScript，JSX Fragment
- [比拼生态和未来，Spark 和 Flink 哪家强？](#)
- [推荐 30 个用于微服务的顶级工具](#)

今天是八月最后一天，本月总共发布了《Spring一周回顾》 [5](#) 篇，Spring 产品发布文章 [15](#) 篇，翻译文章 [3](#) 篇，其中 [2](#) 篇被 CSDN 转载发表，收到不少关注和反馈。如果您对 Spring 相关开源技术感兴趣，可以在头条/微信/微博上搜索 [Spring技术社区](#)，来关注我们。另外，网站即将改版扩充内容，现正在积极招募开源爱好者投稿或者翻译 Spring 项目文档，如有意向请联系我本人，微信号 [rainboyan](#)，暗号 [Spring粉丝](#)。

周末愉快，感谢阅读，我们下周见！

# 2018年8月24日

大家好，Spring 粉丝们，感谢一如既往的等待，又到了每周必读的《Spring一周回顾》时间。9月24-27日，Spring 开发者们最期待的一年一度的 SpringOne Platform 将在 Washington D.C 举办，届时将有众多业界大咖们作为演讲嘉宾。如果您不能前往参加的话，我在这里将会为您带来最前沿的资讯报道。

本周我为您带来了如下内容：

- Spring Integration for AWS 2.0 和 Spring Cloud Stream Kinesis Binder 1.0 正式发布，集成 AWS Kinesis 服务，快速构建一个消息驱动的微服务应用程序。
- Spring Security 5.1.0.RC1 发布，其中针对 OAuth2 的配置和自定义能力增强，继续改进 WebFlux 风格的相关安全配置。
- Spring Boot 2.1.0 M2 发布
- Spring Data Lovelace RC2 发布，优化 JPA 启动速度，众多新特效值得期待！
- Hibernate Validator 6.0.13.Final 发布
- Akka 2.5.15 发布
- Istio 1.0.1 发布
- Gradle 4.10-RC3 发布
- Netty 4.1.29 发布
- Kibana 6.4.0、Logstash 6.4.0 同时发布
- RocksDB 5.14.3 发布
- Flink 1.5.3 发布
- Angular 6.1.4 发布
- Apache Tomcat 8.5.33 和 9.0.11 同时发布，修复
- Apache Struts 2.3.35 和 2.5.17 修复安全问题更新发布，赶紧升级吧！

# 2018年8月17日

大家好，Spring 开发者们，又到了每周必读的《Spring一周回顾》时间。上周翻译的《[Spring Boot最佳实践](#)》非常受欢迎，获得了大量点赞和收藏，同时也被 CSDN 收录并微信转发，相信对于读到的 Spring 开发者们一定会有所帮助，这当然也是我非常乐意为之的事情，感谢大家继续关注！

本周我为你带来了如下内容：

- [Spring Framework 5.1 RC2 发布](#)，继续开发更多新特性和优化工作。
- [Spring Session Bean-M1 和Apple-SR4 同时发布！](#)
- [Spring Tools 4 M14 发布](#)，Spring Tools 4 作为下一代企业级应用开发的 Spring IDE，将会支持 Eclipse Java IDE, Visual Studio Code 和 Atom Editor，极大提高编码效率和改善 Spring 应用的开发体验。
- [Spring Cloud Open Service Broker 2.0.1.RELEASE 已发布](#)
- [Spring Cloud Skipper 1.1.0.M1 发布](#)
- [riff v0.1.1 发布](#)，支持Knative
- [Hibernate ORM 5.3.5.Final 发布](#)，原生支持 Sybase 中的分页查询
- [Hibernate Validator 6.0.12.Final 发布](#)，改进支持 JDK 11
- [Groovy 2.5.2 发布](#)
- [Neo4j 3.4.6 发布](#)，有关新特性的介绍看 [这里](#)
- Spring Data Neo4j 项目负责人和数据传奇人物 Michael Hunger 刚刚宣布 [发布 Neo4j JDBC 驱动程序 3.4.0](#)，支持 Neo4j 3.4.x 中的空间和日期/时间数据类型以及完整的集群/路由支持。
- [继 Kubernetes 之后，Prometheus 正式从 CNCF 毕业](#)
- [WhiteSource 推出免费开源的漏洞检查工具](#)
- [10 种保护 Spring Boot 应用程序的绝佳方法](#)，来自 Matt Raible 和 Simon Maple 合作的成果。
- [16 条 Spring Boot 最佳实践](#)，来自业界 Spring 开发者们的精彩总结！
- 上一期的《[Spring一周回顾](#)》同样精彩，值得一读！

# 2018年8月10日

Hi, Spring 粉丝们，本期一周回顾系列的文章来得稍稍有些延迟，但是相信没有辜负好时光，而等待也是值得的，这是八月的第二个周末。

- Spring Cloud Finchley.SR1 发布，可以开始升级了。
- Spring Cloud Kubernetes 0.3.0 发布，这也是从孵化器毕业后正式进入 Spring Cloud 大家族。Kubernetes 近期也更新发布1.11.2版本
- Spring Cloud for Google Cloud Platform 1.0 正式版发布，这对于使用 GCP 服务的开发团队来讲是个好消息！
- 异步处理基础库 Reactor Californium-M1 在这个夏天发布，未来会被集成到 Spring Boot 2.1.0 中。
- Spring Integration、Spring AMQP、Spring for Apache Kafka、Spring for RabbitMQ 几个项目的维护版本和里程碑版本同时更新发布
- Spring Integration for AWS 2.0.0.RC1 和 Spring Cloud Stream Kinesis Binder 1.0.0.RC1 发布
- Spring Vault 2.1 M1 已发布
- 我翻译了[Spring Boot 最佳实践](#)，并增加了第一条最佳实践，原文在[这里](#)
- 谷歌开源的高性能 RPC 框架 gRPC 1.14.1 发布，如果你需要在 Spring Boot 项目中集成 gRPC，可以参考这个[开源的Starter](#)
- 谷歌开源机器学习库 TensorFlow 1.10.0 正式发布，具体内容见[发布说明](#)。
- Apache ZooKeeper 3.4.13 发布
- 蚂蚁金服开源的基于 Spring Boot 的研发框架 SOFABoot 2.4.4 发布
- Grails 3.3.7 和 3.3.8 接连更新发布，此次更新依赖 Spring Boot 升级到 1.5.15.RELEASE。
- Grails 研发团队发布了微服务框架 MicroNaut 的 1.0.0.M4 版本，升级依赖 Kafka 2.0.0、Netty 4.1.27.Final，支持 Velocity、Thymeleaf 等模板引擎，以及其他改进和 BUG 修复。
- Gradle 4.10 RC1 发布，此次默认启用Java的增量编译功能，此外Kotlin DSL也发布1.0 RC1。
- 面向分布式数据流处理和批量数据处理的开源计算平台 Apache Flink 1.6.0 发布，解决了超过 360 个问题。
- 开源面向文档的数据库管理系统 Apache CouchDB 2.2.0 发布，性能提升以及众多新特性。
- 开源分布式发布订阅消息系统 Apache Pulsar 2.1.0 发布，由 Yahoo 开发，现已捐给 Apache 正在孵化中。
- 开源流处理平台，高性能分布式发布订阅消息系统 Apache Kafka 2.0.0 发布，这里有篇[文章](#)介绍了其中的新特性。
- 开源分布式消息和流数据处理平台 Apache RocketMQ 4.3.0 发布，RocketMQ 联合创始人冯嘉在[文章](#)中详细介绍了 4.3.0 的新特性：支持分布式事务。
- 使用[Trampoline](#)工具来管理本地 Spring Boot 应用实例
- Redis Client Lettuce 4.4.6.Final 和 5.0.5.RELEASE 同时更新发布
- 来自 Netflix 的系统高可用建议
- 一年一度的[SpringOne Platform](#)将9月24-27日在 Washington D.C 举办，届时将有众多业界大咖们作为演讲嘉宾，比如熟知的 Rod Johnson、龙之春、Matt Raible 、Phillip Webb，真想过去现场学习下，见一见几位偶像们，无奈只缺一张机票。
- ASF 的官方全球会议 ApacheCon，将于9月24日-27日在加拿大的蒙特利尔举办，同时也是 Apache 社区庆祝 20 周年的一次盛大活动。
- 由华为团队主导开发的微服务框架 Apache ServiceComb 的 Service-Center 1.0.0 和 ServiceComb

Java-Chassis 1.0.0 发布了

- 如果你是第一次关注这个系列，可以看看[上一期的《Spring 一周回顾》](#)，也一定会有很多收获！

# 2018年8月3日

Hi, Spring 粉丝们，幸苦忙碌了一周，终于又在周五见面了。这是 Spring 开发团队非常忙碌的一周，很多项目都陆续发布了新版本。下面就一一回顾一下本周的成果。

- 首先，Spring 的明星项目 [Spring Boot](#) 相继发布了 [1.5.15](#), [2.0.4](#), 以及下一个非常重要的版本 [2.1.0 的第一个里程碑](#)，该版本基于最新的 Spring Framework 5.1.RC1、Spring Data Lovelace RC1 和 Spring Security 5.1 M2, Undertow 2.0 和 Tomcat 9.0, 支持Servlet 4，更多内容在文章中有详细说明。
- 在过去一个月的时间里，从 [start.spring.io](#) 上创建的项目已经突破了 100 万，这归功于开发团队和社区的积极努力，值得祝贺！
- [Spring Framework 5.0.8](#) 发布
- [Spring Framework 5.1 RC1](#) 发布
- [Spring Security 5.0.7](#) 发布
- [Spring Security 5.1.0 M2](#) 发布
- [Spring Cloud Data Flow 1.6](#) 正式版发布
- [Spring IO Platform Brussels SR12](#) 和 [Brussels-SR12](#) 同时发布
- [Spring Boot Admin 2.0.2](#) 发布，支持 自定义 UI 视图
- [Apache NetBeans 9.0](#) 发布，这里有个 [Spring Boot](#) 的插件可以帮助 Spring Boot 开发
- [Netty 4.1.28](#) 发布
- [Kotlin 1.2.60](#) 发布
- [Keycloak 4.2.0](#) 发布
- [GraalVM 的 1.0 RC5](#) 发布，这里有篇有关 Graal 的译文：[解密新一代 Java JIT 编译器 Graal](#) 可以了解下
- Knative发布之后，有非常多的的关注，这里有个示例：[使用Knative来部署Spring Boot应用到Kubernetes](#)
- Istio 1.0, 已生产就绪！
- 上周日7月29号，[Apache Dubbo & Apache RocketMQ](#)开发者沙龙在深圳科技园空体举办，现场有 400 多人参加，部分 PPT：[展望 Apache RocketMQ5.0 | 谈 RocketMQ 的过去、现在和未来](#), [Apache Dubbo 的开源现状和 2.7 版本规划](#)，在会上又高调宣布开源了 [Nacos](#) 和 [Sentinel](#)，欢迎更多的人积极参与到开源中。演讲文档已经放出，可以从这里 [下载](#)
- GR8Conf US 2018部分演讲文档及代码整理如下：
  - [6 things you need to know about GORM 6](#)
  - [Asynchronous and event-driven Grails applications](#)
  - [Room with a Vue - Building Grails apps with Vue.js](#)
  - [Idiomatic Kotlin in a Java World](#)
  - [From Groovy to Kotlin](#)
  - [High Scalable Streaming Microservices with Kafka Streams](#)
  - [Hands on Ratpack](#)
  - [High performant in-memory datasets with Netflix H0110W, 示例代码](#)

最后，我要隆重介绍下本系列文章的龙大神，其本名 Josh Long，约翰·朗，由于 Long 姓和中文的“龙”的拼音一样，又加上作为 Spring 开发人员和技术布道师，所以他给自己起了个诨号“龙之春”，喜欢自拍，在 Twitter

上很活跃，经常在世界各地参加技术会议。应经常关注本系列文章的女粉们的强烈要求，所以在此公开下他的帅照和微信号： **starbuxman**，对技术男有兴趣的妹子和小姐姐们，可以去撩下他，但是不要说是我给的，;)



Josh 是一名 Java 冠军程序员，5 本书的作者（包括 O'Reilly 即将推出的“云原生 Java：使用 Spring Boot, Spring Cloud 和 Cloud Foundry 设计弹性系统”）和 3 个畅销视频培训（包括与 Phil Webb 合编的“使用 Spring Boot 构建微服务 Livelessons”），以及开源贡献者（Spring Boot, Spring Integration, Spring Cloud, Activiti 和 Vaadin）。这里放一个早期有关他的 [技术演讲的视频](#)，可以一睹他那神乎其神的编写代码的速度，目测手速高达 300 APM。

目前在 Pivotal 公司的 Spring 团队已经工作了 8 年，为此在个人博客上专门写了一篇文章，回忆了过往经历与成长，曾经的“小鲜肉”已变成了“大叔”，:D。

就到这里了，周末愉快，下周见！

# 2018年7月27日

首先，在此恭喜[龙大神](#)被评选进入[2018 最具影响力的Java技术专家](#)，排行榜上面公布了这些专家们的twitter 账号，由于众所周知的原因，Twitter 在中国大陆不能访问，所以你要是想关注的话就只能搭梯子上去了。

好了，各位 Spring 粉丝们，大家周末好！本周新一期的 [This Week in Spring - July 24th](#)如约而至，内容很精彩。

- [Spring Framework 5.1 发布第一个 RC 版本](#)，Spring Boot 2.1 M1 下周发布时将会集成进来，同时 Spring Framework 5.0.8 也已经发布
- [Spring Framework 5.0.8](#)，此版本包括修复了 40 个问题和改进，将会和 Spring Boot 接下来的 2.0.4 版本一起使用
- [Spring Security 5.0.7 发布](#)，关闭了 8 个问题，包括升级依赖和修复 OAuth2 相关问题
- [Spring Data Lovelace RC1 发布](#)，关闭了 194 个 问题
- [Hibernate ORM 5.3.3.Final发布](#)，基于即将到来的 JDK 11 进行兼容测试
- [看看 JDK 11 有哪些特性变更](#)
- [IntelliJ IDEA 2018.2发布了](#)，修复和增强 JDK10/11 的支持，增加了对 Groovy 语言 2.5 和 3.0 的一些支持，内置 Gradle 升级到 4.8，修复 BUG，支持默认任务，自动监测并编译 buildSrc 项目，还有 Spring & Spring Boot 的一些新特性和改进等等
- [Kibana 6.3.2 发布](#)，建议升级
- [Logstash 6.3.2 发布](#)，建议升级
- [HashiCorp Consul 1.2 发布](#)，主要亮点是自动将现有的 Consul 集群转换为服务网格解决方案，同时服务与服务之间的通信自动安全增强，使用 TLS 加密与基于身份的授权
- [Google Releases Knative](#)，Knative 是一个用来构建、部署和管理无服务器工作负载的框架
- [Istio 1.0 发布进入倒计时](#)，准确的说是在7月31号下午三点正式发布
- [Angular 6.1.0 发布了](#)，要使用起来，不能掉队了
- “[应用无忧Spring实战营”活动资料下载](#)
- [Spring 技术社区的指南板块正在改版中](#)，已经发布部分学习教程，后续会继续更新
- [2018 年 5 大微服务发展趋势](#)
- [来自 CNCF 的调查报告：中国的云原生之路](#)
- [GR8Conf US 2018 已经圆满结束](#)，三天的大会干货满满，不能去现场实在遗憾，部分文档和代码已经放出，得赶紧补一补了。

## 2018年7月20日

七月是毕业季令人有些感伤的日子，也是小朋友开始暑假放飞快乐的日子，同时也是一年之中下半场的开始。是的，不知不觉 2018 已经过去了一半，一定有些事没有做好而感到失望，也一定有很多事规划着要做而欣喜。如果你感到我有点情绪低落，那一定是患上了“世界杯后遗症”，据说写文章和跑步可以有效治疗。

各位Spring粉丝们，大家周末好！看着 [龙大神](#)本周又忙着在世界各地参加各种会议活动，为活动准备演讲材料《Reactive Spring》，在忙碌的同时也不忘如期发布本周新一期的 [This Week in Spring](#)。

- [Spring Batch 4.1.0.M2发布了](#)，本版本提供了一些新特性，比如简化分块步骤、新增类 [JsonFileItemWriter](#)、新增了 [BeanValidatingItemProcessor](#)用以支持校验 Bean。
- [Spring REST Docs 1.2.5.RELEASE](#)和[Spring REST Docs 2.0.2.RELEASE](#)同时发布，修复了若干 BUG。[InfoQ](#) 有放出一个关于介绍使用 Spring REST Docs 的[视频](#)。
- [Dependency Management Plugin 1.0.6.RELEASE](#)发布，该版本修复了若干问题，将会包含在随后的 Spring Boot 1.5.15 和 2.0.4 一起提供。
- [Spring Cloud Function](#) 上月已经发布了 1.0.0 正式版本，即将加入到 Spring Initializr 中提供创建项目可选项，再来看看之前有关介绍的 [文章](#)。
- [Spring架构师Eberhard Wolff](#)已经更新了他的演示项目的代码，可以在他的[GitHub主页](#)看到更新。
- [Axon Framework 3.3 提供订阅查询API和支持Kafka集成支持](#)
- [Grails 的研发团队](#)昨天刚刚发布了[MicroNaut 微服务框架](#)的 1.0.0.M3 版本，他们从 Grails 多年的开发中借鉴了大量优秀的设计思想，同时又吸取了之前 Grails 的一些教训，非常期待正式版本的到来。
- [Groovy 2.5.1 发布了](#)，该版本修复了 44 个Bug，为兼容 JDK 多个版本而持续优化。

# 2018年7月13日

“龙之春”**龙大神**每周一期必写的This Week in Spring，相信国内外的Spring 开发者都一定是必读的文章，而我就是其中一个忠实的粉丝。这篇文章不是原文对照翻译，而是摘取一部分再加上国内的热门文章，同时有我的理解和分析，毕竟这是一篇针对中国开发者而写的文章。

最近一段时间在写一本关于 Spring Security 的新书，想着写的过程中先放一部分内容发到博客上，或许能对一部分碰巧遇到同类问题的开发者有所帮助，那也是好的。所以就想到采用 Sagan 的主题来搭建博客，Sagan 的整个代码是采用 Spring Boot 来开发，但对我写个技术文章来讲就显得太复杂了，所以最后采用 JBake [1: JBake 是一款静态网站和博客生成工具，专为开发者和设计师打造，Java 语言开发并且开源] 来构建博客，文章采用 Markdown 或者 AsciiDoc 来编写，最后发布在 Github Pages 上，这样很快就搞定，最重要的是能满足我对写作良好体验的要求。

好了，言归正传。一起来看看，本周 Spring 官方及开发社区发生了哪些新鲜事。

首先来看看 Spring 官方团队给我们带来了

- Spring Tool Suite 3.9.5 版本发布了，升级到 Eclipse Photon，这样就可以支持 JDK 9/10，另外其他一些性能优化和问题修复。有关 Eclipse Photon 的新特性，请到 [这里](#)会有更详细的介绍。
- Spring Cloud Data Flow 1.5.2 发布，同时 1.6 M2 也发布，1.6 的亮点是任务调度原生集成到 PCF。
- Spring Cloud Stream Elmhurst.SR1 发布了，这是一个维护性版本，Spring Boot 默认使用 2.0.3.RELEASE。
- Spring Social 的负责人 Craig Walls 在上周二正式决定 Spring Social 进入 EOL，这包括其中的一些针对特定社交平台的 API 绑定实现项目，比如 Facebook、Twitter、LinkedIn 等，因为其中的一些核心特性已经纳入到 Spring Security 5 了，在 Spring Security 5 中可以很好的扩展来实现达到相应的目的。接下来一年会继续提供问题修复和支持，但是不会发布新特性版本。
- Spring 背后公司的产品 Pivotal Cloud Foundry 2.2 发布
- Spring Boot 项目目前正在招募开源贡献者，用以帮助完善文档和部分功能代码，详情看 [Github 的问题列表](#)。
- Steeltoe.NET 作为 .NET 平台上类似 Spring Cloud 参考实现的云原生应用库，在 Nuget 上已经有超过 50 万的下载量，这背后的云原生技术推手 Pivotal Software 功不可没，开源开放的积极推动旨在吸引和影响更多的开发者加入到这一阵营。

接下来，再看看国内外社区一些新动态

- JHipster 5.1.0 发布，此版本共关闭了 29 个问题和 PR，升级 JHipster Registry 到 v4.0.0，另外修复了 Angular 在构建多个延迟加载模块时出现错误的问题。
- Thibaud Lepretre 最近更新了 cas-security-spring-boot-starter 的 1.0 版本计划，首次支持 Spring Boot 2。
- Netflix 官方在 Eureka 项目 Wiki 上更新：Eureka 2.0 停止开发，但是 1.9.x 仍然会持续更新，这个消息对于朋友圈和社区的一些不明真相的朋友引起了不必要的恐慌。

## 2018年7月6日

Spring 官方团队给我们带来了

- Spring Cloud Data Flow 1.6 M1 和 1.5.2 发布
- Spring Tool Suite 3.9.5 发布
- Spring Social 宣布 EOL
- Spring Social 宣布 EOL
- Spring Cloud Edgware.SR4 发布
- Spring Cloud Open Service Broker 2.0.0.RELEASE 发布
- Spring Cloud Task 1.2.3.RELEASE 发布

Apache 开源社区给我们带来了

- Apache HttpComponents Core 4.4.10 发布
- Apache Kafka 0.11.0.2、0.11.0.3 发布
- Apache Tomcat 7.0.90、8.0.53 发布
- Apache OpenNLP 1.9.0 发布
- Apache CXF 3.2.5、3.1.16 发布

# Spring Framework 发布说明

Spring Framework 目前仍然保持着“一年一个小版本，四年一个大版本”的迭代速度继续前进。[5.0](#) 去年9月28日正式发布，当前最新版本是5.0.11，该版本将会随着Spring Boot 2.0一起终止支持（大概是2019年3月底）。[5.1](#)于今年9月21日正式发布，当前最新版本是5.1.3，这个版本是一个过渡性版本，2019年底将会不再支持。[5.2](#)版本的具体发布时间待定，可能明年年中发布，将会是一个长期支持版本（LTS）版本，一直到2023年底（JDK 8 / 11也是支持到2023年）。

Spring 5.1 主要新功能：

- 默认支持 JDK 8+，支持 JDK 11
- 支持 GraalVM
- 升级 Reactor Core 3.2，Reactor Netty 0.8
- 支持 Hibernate ORM 5.3
- 支持 WebFlux HTTP/2
- 优化内部反射机制，改善启动时间并减少堆内存消耗

Spring 5.0 主要新功能：

- 默认支持 JDK 8+，兼容 JDK 9
- 支持 Java EE 7+，兼容 Java EE 8
- 新的模块[spring-webflux](#)，响应式风格的，基于 Netty 和 Undertow 实现
- 支持 Kotlin 语言
- 测试升级到 JUnit 5

表格 2. Spring Framework 2018 年版本发布记录

版本	发布时间	发布说明
5.1.3.RELEASE	2018-11-27 09:28:52	Spring Framework 5.1.3 发布
5.0.11.RELEASE	2018-11-27 08:54:00	Spring Framework 5.0.11 发布
4.3.21.RELEASE	2018-11-27 07:39:58	Spring Framework 4.3.21 发布
5.1.2.RELEASE	2018-10-29 10:33:14	Spring Framework 5.1.2 发布
5.1.1.RELEASE	2018-10-15 07:20:29	Spring Framework 5.1.1 发布
5.0.10.RELEASE	2018-10-15 08:01:50	Spring Framework 5.0.10 发布
4.3.20.RELEASE	2018-10-15 08:48:13	Spring Framework 4.3.20 发布
5.1.0.RELEASE	2018-09-21 07:26:25	Spring Framework 5.1.0 发布
5.1.0.RC3	2018-09-07 13:28:58	Spring Framework 5.1 RC3 发布
5.0.9.RELEASE	2018-09-07 12:15:56	Spring Framework 5.0.9 发布
4.3.19.RELEASE	2018-09-07 14:09:54	Spring Framework 4.3.19 发布
5.1.0.RC2	2018-08-17 09:27:02	
5.1.0.RC1	2018-07-26 07:42:11	
5.0.8.RELEASE	2018-07-26 07:49:30	
5.0.7.RELEASE	2018-06-12 15:09:44	
4.3.18.RELEASE	2018-06-12 14:49:42	
5.0.6.RELEASE	2018-05-08 08:34:16	
4.3.17.RELEASE	2018-05-08 07:48:24	
4.3.16.RELEASE	2018-04-09 14:57:45	
5.0.5.RELEASE	2018-04-03 20:11:20	
4.3.15.RELEASE	2018-04-03 20:10:35	
5.0.4.RELEASE	2018-02-19 11:12:47	
5.0.3.RELEASE	2018-01-23 09:42:24	
4.3.14.RELEASE	2018-01-23 09:03:37	

# Spring Framework 5.1.3 发布

Spring Framework 5.1.3 发布，这个版本是一些问题修复。

5.1.3 是 5.1 之后的第三个维护性版本，解决了 超过 45 个问题单。

Spring Boot 2.1 的第一个维护升级版本，基于 Spring Framework 5.1.3，即将推出！开发团队还将在本周晚些时候跟进相应的 Spring Boot 2.0.7 和 1.5.18 版本。

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Framework 5.1.2 发布

Spring Framework 5.1.2 发布，该版本修复了一些问题。

5.1.2 是 5.1 之后的第二个维护性版本，解决了 30 个问题单。

Spring Boot 2.1.x 在发布了多个里程碑版本和候选（RC）版本之后，将会在本周发布 2.1.0 正式版，基于最新的 Spring Framework 5.1.2。

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Framework 5.1.1 发布

Spring Framework 5.1.1 发布，这个版本是一些问题修复。

[5.1.1](#) 是 5.1 之后的第一个维护性版本，解决了 [30 多个问题单](#)。

Spring Boot 2.1 的第一个候选版本，基于 Spring Framework 5.1.1，即将推出！开发团队还将在本周晚些时候跟进相应的 Spring Boot 2.0.6 和 1.5.17 版本。

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Framework 5.1.0 发布

Juergen Hoeller 宣布了，Spring Framework 5.1 正式发布，可以从 [repo.spring.io](#) 以及 Maven Central 获取到。

Spring Framework 5.1 需要 JDK 8 及以上版本，特别是支持 JDK 11 这个 LTS 版本。因此推荐升级到 5.1 版本，以便能够使用 JDK 11 的新特性。除此之外，该版本还支持 [GraalVM](#)，这是 Oracle 四月份宣布开源的一个支持多语言的高性能虚拟机，目前在开源社区已经大量测试使用。

Spring Framework 5.1 升级了若干依赖：[Reactor Californium](#)、[Hibernate ORM 5.3](#) 和 [JUnit 5.3](#)。

核心容器为 Java 和 Kotlin 引入了功能 bean 定义改进，包括功能 bean 检索方式。Spring 对内部反射的使用进行了优化，以改善启动时间并减少堆内存消耗。

Web 应用程序栈提供从端点到核心容器的人性化调试日志体验。它具有用于功能 Web 端点的 DSL 样式构建器，并将 WebFlux HTTP/2 支持产品扩展到 Netty 运行时。

Spring Boot 2.1 M4 将会在下周发布，这样就可以在 [Spring Initializr](#) 使用到 Spring Framework 5.1 GA！即将于 10 月中旬推出的 Spring Boot 2.1 RC1 预计将针对 Spring Framework 5.1.1 发布，请继续关注。

更多地新特性，请看这里 [What's New in Spring Framework 5.1](#)，以及 [如何升级到 Spring Framework 5.1](#)，还可以查看 [参考手册](#)。

# Spring Framework 5.1 RC3 发布

Spring Framework 5.1 RC3 修复了 30 个问题单。

接下来，Spring Boot 2.1 M3 将会在下周发布，敬请期待！

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Framework 5.0.11 发布

Spring Framework 5.0.11 发布，这个版本是一些问题修复。

[5.0.11](#) 也解决了 34 个问题单。

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Framework 5.0.10 发布

Spring Framework 5.0.10 发布，这个版本是一些问题修复。

5.0.10 解决了 20多个问题单。

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Framework 5.0.9 发布

Spring Framework 5.0.9 修复了 36 个问题单。

接下来，Spring Boot 2.0.5 将会在下周发布，敬请期待！

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Framework 4.3.21 发布

Spring Framework 4.3.21 发布，这个版本是一些问题修复。

[4.3.21](#) 则是一个相当小的补丁发布。

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Framework 4.3.20 发布

Spring Framework 4.3.20 发布，这个版本是一些问题修复。

4.3.20 是一个相当小的补丁发布。

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Framework 4.3.19 发布

Spring Framework 4.3.19 修复了 **23** 个问题单。

接下来，Spring Boot 1.5.16 将会在下周发布，敬请期待！

[项目主页](#) | [GitHub](#) | [问题](#) | [文档](#)

# Spring Boot 发布说明

Spring Boot 基于“约定优于配置”，可以轻松、快速的创建独立的、生产级的基于 Spring 的应用程序，并可以立即运行起来。这使得开发者可以更加专注于开发，而无需为各种常规的样板配置而烦恼。

其特性如下：

- 创建独立的 Spring 应用程序
- 直接嵌入 Tomcat、Jetty 或 Undertow（无需部署WAR文件）
- 提供可自由选择的“起步”依赖项以简化构建配置
- 尽可能自动配置 Spring 和第三方库
- 提供生产就绪功能，例如指标，运行状况检查和外部化配置
- 绝对没有代码生成，也不需要 XML 配置

Spring Boot 自 2014 年 4 月 1 日正式发布 1.0 版本，2018 年 3 月 1 日发布了 2.0.0，目前最新的版本是 2.1.1。

表格 3. Spring Boot 2018 年版本发布记录

版本	发布时间	发布说明
2.1.1.RELEASE	2018-11-30 10:01:10	Spring Boot 2.1.1 发布
2.0.7.RELEASE	2018-11-30 04:52:58	Spring Boot 2.0.7 发布
1.5.18.RELEASE	2018-11-29 11:34:42	Spring Boot 1.5.18 发布
2.1.0.RELEASE	2018-10-30 10:50:21	Spring Boot 2.1.0 正式发布
2.1.0.RC1	2018-10-17 15:30:37	
2.0.6.RELEASE	2018-10-16 18:01:52	Spring Boot 2.0.6 发布
1.5.17.RELEASE	2018-10-16 10:19:28	Spring Boot 1.5.17 发布
2.1.0.M4	2018-09-25 01:21:58	
2.1.0.M3	2018-09-13 08:19:53	
2.0.5.RELEASE	2018-09-12 11:18:00	Spring Boot 2.0.5 发布
1.5.16.RELEASE	2018-09-11 16:17:43	Spring Boot 1.5.16 发布
2.1.0.M2	2018-08-21 11:13:05	
2.1.0.M1	2018-07-31 09:18:29	
2.0.4.RELEASE	2018-07-31 09:17:21	Spring Boot 2.0.4 发布
1.5.15.RELEASE	2018-07-31 09:16:03	Spring Boot 1.5.15 发布
2.0.3.RELEASE	2018-06-14 21:15:01	
1.5.14.RELEASE	2018-06-14 10:05:15	
2.0.2.RELEASE	2018-05-09 18:26:53	
1.5.13.RELEASE	2018-05-09 09:31:33	
1.5.12.RELEASE	2018-04-09 23:14:20	
2.0.1.RELEASE	2018-04-05 11:32:25	
1.5.11.RELEASE	2018-04-05 09:01:38	
2.0.0.RELEASE	2018-03-01 04:42:15	
2.0.0.RC2	2018-02-21 12:28:45	
2.0.0.RC1	2018-01-31 05:47:02	
1.5.10.RELEASE	2018-01-30 23:29:22	

# Spring Boot 2.1.1 发布

Spring Boot 2.1.1 已经发布，共解决了 70 项问题修复、改进和依赖项更新，建议使用 2.1.x 的用户升级到此版本。

其中依赖升级到最新版本：

- Spring AMQP 2.1.2.RELEASE
- Spring Data Lovelace-SR3
- Spring Framework 5.1.3
- Spring Integration 5.1.1.RELEASE
- Spring Kafka 2.2.2.RELEASE
- Spring Security 5.1.2.RELEASE
- Spring Session Bean-SR1
- Byte Buddy 1.9.5
- Couchbase Client 2.7.1
- Ehcache3 3.6.2
- Elasticsearch 6.4.3
- Flyway 5.2.3
- Groovy 2.5.4
- Infinispan 9.4.3.Final
- Jetty Reactive HttpClient 1.0.2
- Jooq 3.11.7
- Junit Jupiter 5.3.2
- Kafka 2.0.1
- Lettuce 5.1.3.RELEASE
- Lombok 1.18.4
- Micrometer 1.1.1
- Mockito 2.23.4
- Netty Tcnative 2.0.20.Final
- Reactor Californium-SR3
- Rxjava2 2.2.4
- Tomcat 9.0.13
- Undertow 2.0.16.Final

# Spring Boot 2.1.0 正式发布

Andy Wilkinson 在博客宣布了 Spring Boot 2.1.0 发布。

现在已经可以从 [Maven Central](#), [Bintray](#), 和 [Spring release](#)仓库下载到。

此版本增加了大量新功能和改进。有关完整 [升级说明](#)以及 新的和值得注意的功能, 请参阅 [发行说明](#)。

## 2.1 中的新功能

### 第三方库升级

我们尽可能升级到其他第三方库的最新稳定版本。此版本中的一些值得注意的依赖项升级包括：

- Hibernate 5.3
- Micrometer 1.1
- Reactor Californium
- Spring Data Lovelace
- Spring Framework 5.1
- Tomcat 9
- Undertow 2

### 性能改进

作为我们不断努力提高性能的一部分, 我们在 Spring Boot 2.1 中取得了一些重大进展。应用程序现在可以更快地启动并消耗更少的内存。这在具有非常严格的内存限制的环境中尤其有用。

我们还接受了 Spring Framework 和 Spring Data JPA 对 Hibernate 异步启动的支持。如果您使用 Spring Data JPA 并设置 `spring.data.jpa.repositories.bootstrap-mode=deferred`, 则 Hibernate 将在单独的线程中启动, 而应用程序的其余启动处理将并行进行。

### 支持 Java 11

继 Spring Framework 5.1 对 Java 11 的支持之后, Spring Boot 2.1 现在也支持 Java 11, 同时还与 Java 8 保持兼容。

### 支持 DataSize

如果属性需要以字节或类似方便的单位表示大小, 则它可以设置 `org.springframework.util.unit.DataSize` 属性。与 Spring Boot 2.0 中引入的 `Duration` 支持类似, 数据大小支持允许在 `application.properties` 中配置值时指定单位。例如, `10MB` 可用于 10 兆字节的值。

### Actuator 端点

Spring Boot 2.1 中引入了两个新的 Actuator 端点：

- `/actuator/caches` 提供有关应用程序缓存管理器的信息
- `/actuator/integrationgraph` 提供了 Spring Integration 组件的图形表示

还增强了 `health` 端点, 以允许对单个组件的健康状况进行查询。例如, 对 `/actuator/health/db` 的请求仅执行 “db” `HealthIndicator`。

## Metrics

除了升级到 Micrometer 1.1 之外，还添加了用于导出到 AppOptics、Humio 和 KariosDB 的自动配置。指标范围也得到了改进，包括：

- Hibernate 度量
- Spring Framework 的 WebClient
- Kafka 消费度量
- Log4j2 度量
- Jetty 服务器线程池度量
- 服务端 Jersey HTTP 请求度量

发行说明中记录了许多其他变更和改进。您还可以在下一版本中找到我们计划删除的已弃用类和方法的列表。

## 致谢

我们想借此机会再次感谢所有用户和贡献者。我们现在有超过 500 人提交代码，并且已经有超过 19000 个提交到该项目。

如果您有兴趣帮忙，请查看问题库中的“ideal for contribution”标签。如果您有一般性问题，请在 stackoverflow.com 上使用 spring-boot 标签或在 Gitter 上与社区其他人聊天。

[项目主页](#) | [GitHub](#) | [Issues](#), [使用文档](#)

NOTE

原文：<https://spring.io/blog/2018/10/30/spring-boot-2-1-0>  
作者：Andy Wilkinson

# Spring Boot 2.0.7 发布

Spring Boot 2.0.7 已经发布，共解决了 81 项问题修复、改进和依赖项更新，建议使用 2.0.x 的用户升级到此版本。

其中依赖升级到最新版本：

- Spring AMQP 2.0.10.RELEASE
- Spring Cloud Connectors 2.0.4.RELEASE
- Spring Data Kay SR12
- Spring Framework 5.0.11
- Spring Integration 5.0.10
- Spring Kafka 2.1.11.RELEASE
- Spring Security 5.0.10
- Spring Session Apple-SR7
- Activemq 5.15.8
- Ehcache 2.10.6
- Elasticsearch 5.6.13
- Janino 3.0.11
- Javax Json 1.1.4
- Jetty El 8.5.33.1
- Kotlin 1.2.71
- Micrometer 1.0.8
- Neo4j Ogm 3.1.5
- Netty 4.1.31.Final
- Reactor Bom Bismuth-SR14
- Thymeleaf 3.0.11.RELEASE
- Thymeleaf Extras Java8time 3.0.2.RELEASE
- Thymeleaf Extras Springsecurity4 3.0.4.RELEASE
- Unboundid Ldapsdk 4.0.9

# Spring Boot 2.0.6 发布

Spring Boot 2.0.6 发布，共解决了 97 项问题修复、改进和依赖项更新和 2 个安全问题，建议使用 2.0.x 的用户升级到此版本。

其中部分新特性：

- 弃用 `SecurityPrerequisite`
- 为 Thymeleaf 的 Spring Security 5 dialect 提供自动配置，并弃用 Spring Security 4 dialect 的自动配置
- 允许 `@ConditionalOnEnabledEndpoint` 用于任何组件
- 为了更好地与 Flyway 5.1 兼容，请避免在 `flywayCallbacks` 为空时调用 `setCallbacks`
- 使用 `ApplicationContextRunner` 时，可以更轻松地显示条件评估报告

依赖升级到最新版本：

- Spring Framework 5.0.10.RELEASE
- Spring Data Kay SR11
- Spring Security 5.0.9.RELEASE
- Spring Session Apple-SR6
- Spring Integration 5.0.9
- Spring AMQP 2.0.8.RELEASE
- Spring Web Services 3.0.4.RELEASE
- Reactor Bismuth-SR12
- Micrometer 1.0.7
- Thymeleaf 3.0.10.RELEASE
- Thymeleaf Extras Spring Security 3.0.3.RELEASE
- Ehcache3 3.5.3
- Neo4j Ogm 3.1.4
- Rabbit Amqp Client 5.4.3
- Hibernate Validator 6.0.13.Final
- Elasticsearch 5.6.12
- Jackson 2.9.7
- Janino 3.0.10
- Javax Jaxb 2.3.1
- Javax Json 1.1.3
- Johnzon Jsonb 1.1.10
- Unboundid Ldapsdk 4.0.8

2.0.5 的发布记录在 [这里](#) 查看。

# Spring Boot 2.0.5 发布

Spring Boot 2.0.5 发布，共解决了 89 项问题修复、改进和依赖项更新，建议使用 2.0.x 的用户升级到此版本。

其中部分新特性：

- 增加条件到 `WebServicesAutoConfiguration` 以支持 lists
- WebFlux 和 WebMvc 增加支持 `MeterFilter#maximumAllowableTags`
- WebFlux 自动配置 `HiddenHttpMethodFilter`
- 允许 actuator 端点被 `MvcMatchers` 使用

其中修复已知 BUG：

- `EmbeddedWebServerFactoryCustomizerAutoConfiguration` 缺少注解 `@ConditionalOnWebApplication`
- `ReactiveSecurityAutoConfiguration` 缺少条件注解
- Spring Integration 的 JDBC 自动配置没有使用自动配置的 `DataSource`
- `EndpointServerWebExchangeMatcher.matches` 出现 NPE
- `OriginTrackedFieldError.toString()` 被调用时会出现堆栈溢出
- `FlywayEndpoint` 的基线 schema 出现 NPE
- 依赖配置中缺少 `micrometer-registry-cloudwatch`
- 当使用 `starter-web`, `starter-webflux` 和 `spring.main.web-application-type=reactive` 时会出现 `NoSuchBeanDefinitionException`
- `HttpExchangeTracer#postProcessRequestHeaders` 从不被调用
- 当 Couchbase 消失时，`CouchbaseHealthIndicator` 可以挂起
- Spring Boot 2.x 中 `localProperties` 优先级高于 `environmentProperties`

依赖升级到最新版本：

- Spring Framework 5.0.9
- Spring Data Kay SR10
- Spring Session Apple-SR5
- Spring Security 5.0.8.RELEASE
- Spring Integration 5.0.8
- Spring Kafka 2.1.10.RELEASE
- Spring Amqp 2.0.6.RELEASE
- Spring Kafka 2.1.9.RELEASE
- Spring Cloud Connectors 2.0.3.RELEASE
- ActiveMQ 5.15.6
- Derby 10.14.2.0
- Elasticsearch 5.6.11
- Hibernate Validator 6.0.12.Final

- Httpasyncclient 4.1.4
- Infinispan 9.3.3.Final
- Janino 3.0.9
- Jaybird 3.0.5
- Javax Mail 1.6.2
- Jest 5.3.4
- Jetty El [8.5.33](#)
- Johnzon Jsonb 1.1.9
- Junit Jupiter 5.3.1
- Lettuce 5.0.5.RELEASE
- Neo4j Ogm [3.1.2](#)
- Netty [4.1.29.Final](#)
- Postgresql 42.2.5
- Reactor Bom [Bismuth-SR11](#)
- Rxjava2 2.1.17
- Unboundid Ldapsdk 4.0.7

相关文章：

- [Spring Boot 2.0.4 发布查看。](#)

# Spring Boot 2.0.4 发布

Spring Boot 2.0.4 包含了 90 个修复、改进和依赖项更新。

其中修复的部分BUG

- 当使用 EndpointRequest.toAnyEndpoint() 时 /actuator/ 不安全
- 提供一致的方法来发现主要 DispatcherServlet 的路径
- 项目特定设置未在 Eclipse 中应用
- 由于缺少默认构造函数而无法绑定的属性很难诊断
- 在使用 JdbcTemplate 之前，数据库迁移可能尚未运行
- RedisCacheConfiguration 缺少 @ConditionalOnClass 检查

依赖升级

- Spring Framework 5.0.8
- Spring Data Kay SR9
- Spring Security 5.0.7.RELEASE
- Spring Session Apple-SR4
- Spring Integration 5.0.7.RELEASE
- Spring Web Services 3.0.3.RELEASE
- Spring Kafka 2.1.8.RELEASE
- Spring HATEOAS 0.25.0.RELEASE
- Spring Amqp 2.0.5.RELEASE
- Spring REST Docs 2.0.2.RELEASE
- Micrometer 1.0.6
- Netty 4.1.27.Final
- Hibernate Validator 6.0.11.Final
- Kafka 1.0.2
- Jna 4.5.2
- Mariadb 2.2.6
- Postgresql 42.2.4
- Kotlin 1.2.51
- Solr 6.6.5
- Johnzon Jsonb 1.1.8
- Jooq 3.10.8
- Rxjava2 2.1.16

# Spring Boot 1.5.18 发布

Spring Boot 1.5.18 已经发布，共解决了 16 项问题修复、改进和依赖项更新。

需要说明的是，由于 Spring Boot 1.x 已经 [宣布进入 EOL 了](#)，如果您还没有升级，请尽快考虑升级和迁移。

第三方依赖升级到最新版本：

- Spring Framework 4.3.21.RELEASE
- Spring Data Ingalls SR17
- Spring Amqp 1.7.11.RELEASE
- Spring Security 4.2.10.RELEASE
- Spring Security OAuth 2.0.16
- Spring Session 1.3.4.RELEASE
- Spring Web Services 2.4.3.RELEASE
- Spring Integration 4.3.18.RELEASE
- Spring Cloud Connectors 1.2.7.RELEASE
- Jackson 2.8.11.20181123
- Appengine Sdk 1.9.67
- Tomcat 8.5.35

# Spring Boot 1.5.17 发布

Spring Boot 1.5.17 发布，共解决了 19 项问题修复、改进和依赖项更新，建议升级。

第三方依赖升级到最新版本：

- Spring Framework 4.3.20.RELEASE
- Spring Data Ingalls SR16
- Spring Amqp 1.7.11.RELEASE
- Spring Security 4.2.9.RELEASE
- Spring Security OAuth 2.0.16
- Spring Web Services 2.4.3.RELEASE
- Rabbit Amqp Client 4.8.3
- Appengine Sdk 1.9.66
- GemFire 8.2.12
- Undertow 1.4.26.Final

## Spring Boot 1.5.16 发布

Spring Boot 1.5.16 发布，共解决了 27 项问题修复、改进和依赖项更新，建议升级。

依赖升级到最新版本：

- Spring Framework 4.3.19
- Spring Security 4.2.8.RELEASE
- Spring Data Ingalls SR15
- Rabbit Amqp Client 4.8.1
- Spring Amqp 1.7.10
- Httpasyncclient 4.1.4
- Jetty 9.4.12.v20180830
- Tomcat 8.5.34
- Mysql 5.1.47

上次发布记录在[这里](#)。

# Spring Boot 1.5.15 发布

Spring Boot 1.5.15 包含了 35 个修复、改进和依赖项更新。

其中修复的部分 BUG

- Flyway 文件系统前缀位置检查
- 当 Jetty 的 WebApplicationContext 无法启动时，应用程序仍然启动成功
- 由于 BeanTypeRegistry 持有过时信息，重写的 bean 定义会导致错误的 Bean 条件判断
- 在 springProfile 的 name 属性中使用时，不会替换属性占位符
- 自动配置的 MultipartConfigElement 会阻止 CommonsMultipartResolver 解析请求部分
- 如果 Undertow 启用了访问日志记录，当容器停止时，线程会泄漏

依赖升级

- Spring Data Ingalls SR14
- Spring Amqp 1.7.9.RELEASE
- Dependency Management Plugin 1.0.6.RELEASE
- Httpclient 4.5.6
- Httpcore 4.4.10
- Tomcat 8.5.32
- Narayana 5.5.32.Final
- Maven Enforcer Plugin 1.4.1
- Git Commit Id Plugin 2.2.4

# Spring Data 发布说明

Spring Data 的使命是为数据访问提供熟悉且一致的基于 Spring 的编程模型，同时仍保留底层数据存储的特殊特性。

它使得使用数据访问技术，关系数据库和非关系数据库，map-reduce 框架和基于云的数据服务变得容易。这是一个伞形项目，其中包含许多特定于给定数据库的子项目。这些项目是通过与这些令人兴奋的技术背后的许多公司和开发人员合作开发的。

其特性如下：

- 强大的存储库和自定义对象映射抽象
- 从存储库方法名称派生动态查询
- 实现域基类提供基本属性
- 支持透明审核（创建，最后更改）
- 可以集成自定义存储库代码
- 通过 JavaConfig 和自定义 XML 命名空间轻松实现 Spring 集成
- 与 Spring MVC 控制器的高级集成
- 实验性支持跨存储持久性

主要项目：

- Spring Data Commons
- Spring Data JDBC
- Spring Data JPA
- Spring Data KeyValue
- Spring Data LDAP
- Spring Data MongoDB
- Spring Data Redis
- Spring Data REST
- Spring Data for Apache Cassandra
- Spring Data for Apache Geode
- Spring Data for Apache Solr
- Spring Data for Pivotal GemFire

## Spring Data 发布说明

版本	发布时间	发布说明
2.1.3.RELEASE	2018-11-27 13:23:36	Spring Data Lovelace SR3 发布
2.0.12.RELEASE	2018-11-27 11:16:35	Spring Data Kay SR12 发布
1.9.17.RELEASE	2018-11-27 10:10:59	Spring Data Ingalls SR17 发布
2.1.2.RELEASE	2018-10-29 12:59:19	Spring Data Lovelace SR2 发布
2.1.1.RELEASE	2018-10-15 09:12:15	Spring Data Lovelace SR1 发布
2.0.11.RELEASE	2018-10-15 10:28:14	Spring Data Kay SR11 发布
1.9.16.RELEASE	2018-10-15 11:55:51	Spring Data Ingalls SR16 发布
2.1.0.RELEASE	2018-09-21 11:45:29	Spring Data Lovelace 正式发布
2.0.10.RELEASE	2018-09-10 11:52:29	Spring Data Kay SR10 发布
1.9.15.RELEASE	2018-09-10 08:01:44	Spring Data Ingalls SR15 发布
2.1.0.RC2	2018-08-20 08:56:53	
1.9.14.RELEASE	2018-07-27 09:09:43	
2.1.0.RC1	2018-07-26 10:32:30	
2.0.9.RELEASE	2018-07-26 13:23:25	
2.0.8.RELEASE	2018-06-13 19:24:36	
1.9.13.RELEASE	2018-06-13 12:42:36	
2.1.0.M3	2018-05-17 08:09:36	
2.0.7.RELEASE	2018-05-08 13:04:29	
1.9.12.RELEASE	2018-05-08 09:56:09	
2.1.0.M2	2018-04-13 13:08:35	
2.0.6.RELEASE	2018-04-04 14:42:35	
1.9.11.RELEASE	2018-04-04 12:58:04	
2.0.5.RELEASE	2018-02-28 09:43:36	
2.0.4.RELEASE	2018-02-19 19:29:09	
2.1.0.M1	2018-02-06 09:11:17	
2.0.3.RELEASE	2018-01-24 12:46:11	
1.9.10.RELEASE	2018-01-24 10:40:31	

# Spring Data Lovelace SR3 发布

Spring Data 团队宣布发布 Lovelace SR3 维护版本。这个版本获取最新的 Spring Framework 维护版本：5.1.3。Spring Boot 接下来发布的版本（2.1.1）将选取 Lovelace SR3 以便于使用。

该服务发布主要包含错误修复和一些依赖性升级，总共修复了 100 多个问题。您可以通过以下链接找到完整的问题列表：

- [Lovelace SR3](#)

为了解决这些问题，以下是各个更改日志、文档和工件的链接：

## Lovelace SR3

- [Spring Data Commons 2.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data JPA 2.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data MongoDB 2.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data KeyValue 2.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Solr 4.0.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Gemfire 2.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Neo4j 5.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Cassandra 2.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Geode 2.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data LDAP 2.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Envers 2.1.3 - Artifacts - Javadoc - Documentation](#)
- [Spring Data REST 3.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Redis 2.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Elasticsearch 3.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Couchbase 3.1.3 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data JDBC 1.0.3 - Artifacts - Javadoc - Documentation - Changelog](#)

# Spring Data Lovelace SR2 发布

Spring Data 团队宣布 **Lovelace SR2** 版本发布，这个版本基于刚刚发布的 [Spring Framework 5.1.2](#)之上构建，Lovelace SR2 将会被选入接下来几天就会发布的 Spring Boot 2.1.0 GA 版本中。如果你在使用 Spring Boot 2.0.x，并且想使用这个 SR 版本，那么你可以设置版本属性 (`spring-data-releasetrain.version`) 为 **Lovelace-SR2**。

这个 SR 发布主要包含错误修复和一些依赖性升级，总共修复了 [30个问题](#)。

为了解决这些问题，以下是各个更改日志、文档和工件的链接：

- [Spring Data Commons 2.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data JPA 2.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data MongoDB 2.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Solr 4.0.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Cassandra 2.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data KeyValue 2.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Gemfire 2.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Neo4j 5.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Geode 2.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data LDAP 2.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Envers 2.1.2 - Artifacts - Javadoc - Documentation](#)
- [Spring Data REST 3.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Redis 2.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Elasticsearch 3.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Couchbase 3.1.2 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data JDBC 1.0.2 - Artifacts - Javadoc - Documentation - Changelog](#)

# Spring Data Lovelace SR1 发布

Spring Data 团队宣布发布 Lovelace SR1 维护版本。这个版本获取最新的 Spring Framework 维护版本：5.1.1。

该服务发布主要包含错误修复和一些依赖性升级，总共修复了 70 多个问题。您可以通过以下链接找到完整的问题列表：

- [Lovelace SR1](#)

为了解决这些问题，以下是各个更改日志、文档和工件的链接：

## Lovelace SR1

- [Spring Data Commons 2.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data JPA 2.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Solr 4.0.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data MongoDB 2.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data KeyValue 2.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Cassandra 2.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Gemfire 2.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Neo4j 5.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Geode 2.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data LDAP 2.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Envers 2.1.1 - Artifacts - Javadoc - Documentation](#)
- [Spring Data REST 3.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Redis 2.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Elasticsearch 3.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Couchbase 3.1.1 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data JDBC 1.0.1 - Artifacts - Javadoc - Documentation - Changelog](#)

# Spring Data Lovelace 正式发布

Spring Data 的项目开发者 [Mark Paluch](#) 在博客宣布了 [Spring Data Lovelace 正式发布](#)，本次大版本一共关闭了 [936 个问题单](#)。这个版本的火车构建在刚刚发布的 Spring Framework 5.1 GA 之上。您可以在下周的 Spring Boot 2.1 M4 版本中轻松使用 Spring Data Lovelace。Spring Data Lovelace 附带了许多主要功能，改进和错误修正。

最值得注意的主题有：

- 支持不可变对象
- 延迟 JPA 存储库初始化
- 支持 MongoDB 4.0 客户端会话和事务
- 新的 Spring Data JDBC 模块
- Apache Cassandra 映射对 Map 和元组类型，Lifecycle Callbacks 和 Kotlin Extensions 的改进
- 使用 Spring Data Redis 读取副本

该版本将在即将到来的 Spring Boot 里程碑版本中自动获取。如果要将 Spring Boot 2.0 项目升级到 Lovelace，只需将 `spring-data-releasetrain.version` 属性设置为 [Lovelace-RELEASE](#) 即可。请关注接下来的博客文章，我们将详细地解释最显著的更新。您可以在我们的 JIRA 中找到 [所有问题](#) 的完整列表。

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.data</groupId>
      <artifactId>spring-data-releasetrain</artifactId>
      <version>Lovelace-RELEASE</version>
      <scope>import</scope>
      <type>pom</type>
    </dependency>
  </dependencies>
</dependencyManagement>
```

开发团队已经规划了下一轮发布计划：Moore（摩尔），以 [Gordon Moore](#)（戈登·摩尔）命名，已经开始。

以下是更改日志，文档和工件的链接：

- Spring Data Commons 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data JPA 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data MongoDB 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data KeyValue 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Gemfire 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Neo4j 5.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)

- Spring Data for Apache Cassandra 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Solr 4.0 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Geode 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data LDAP 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Envers 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#)
- Spring Data REST 3.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Redis 2.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Elasticsearch 3.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Couchbase 3.1 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data JDBC 1.0 GA - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)

# Spring Data Kay SR12 发布

Spring Data 团队宣布发布 Kay SR12 维护版本。这个版本获取最新的 Spring Framework 维护版本：5.0.11。Spring Boot 接下来发布的版本（2.0.7）将选取 Kay SR12 以便于使用。

该服务发布主要包含错误修复和一些依赖性升级，总共修复了 100 多个问题。您可以通过以下链接找到完整的问题列表：

- [Kay SR12](#)

为了解决这些问题，以下是各个更改日志、文档和工件的链接：

## Kay SR12

- Spring Data Commons 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data JPA 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data MongoDB 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data KeyValue 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Solr 3.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Gemfire 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Neo4j 5.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Cassandra 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Geode 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data LDAP 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Envers 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#)
- Spring Data REST 3.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Redis 2.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Elasticsearch 3.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Couchbase 3.0.12 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)

# Spring Data Kay SR11 发布

Spring Data 团队宣布发布 Kay SR11 维护版本。这个版本获取最新的 Spring Framework 维护版本：5.0.10。Spring Boot 将通过其 Spring Boot 2.0.6 版本选取 Kay SR11 以便于使用。

该服务发布主要包含错误修复和一些依赖性升级，总共修复了 70 多个问题。您可以通过以下链接找到完整的问题列表：

- [Kay SR11](#)

为了解决这些问题，以下是各个更改日志、文档和工件的链接：

## Kay SR10

- [Spring Data Commons 2.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data JPA 2.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Solr 3.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data MongoDB 2.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data KeyValue 2.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Cassandra 2.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Gemfire 2.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Neo4j 5.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data for Apache Geode 2.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data LDAP 2.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Envers 2.0.11 - Artifacts - Javadoc - Documentation](#)
- [Spring Data REST 3.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Redis 2.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Elasticsearch 3.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)
- [Spring Data Couchbase 3.0.11 - Artifacts - Javadoc - Documentation - Changelog](#)

# Spring Data Kay SR10 发布

Spring Data Kay SR10 发布，属于维护更新版本。

Kay SR10 在刚发布的 Spring Framework 5.0.9 之上发布，Spring Boot 2.0.5 将会更新到此版本。

该版本是修复了一些 Bug 和升级一些第三方依赖，具体来说，Kay SR10 关闭了 75 问题单。

## Kay SR10

- Spring Data Commons 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data JPA 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data KeyValue 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Solr 3.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Gemfire 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Neo4j 5.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data MongoDB 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Cassandra 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Geode 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data LDAP 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Envers 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#)
- Spring Data REST 3.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Redis 2.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Elasticsearch 3.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Couchbase 3.0.10 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)

# Spring Data Ingalls SR17 发布

Spring Data 团队宣布发布 Ingalls SR17 等维护版本。这个版本获取最新的 Spring Framework 维护版本：4.3.21。Spring Boot 接下来发布的版本（1.5.18）将选取 Ingalls SR17 以便于使用。

该服务发布主要包含错误修复和一些依赖性升级，总共修复了 100 多个问题。您可以通过以下链接找到完整的问题列表：

- Ingalls SR17

为了解决这些问题，以下是各个更改日志、文档和工件的链接：

## Ingalls SR17

- Spring Data Commons 1.13.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data JPA 1.11.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data MongoDB 1.10.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data KeyValue 1.2.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Solr 2.1.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Gemfire 1.9.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Neo4j 4.2.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Cassandra 1.5.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data LDAP 1.0.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Envers 1.1.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#)
- Spring Data REST 2.6.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Redis 1.8.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Elasticsearch 2.1.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Couchbase 2.2.17 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)

# Spring Data Ingalls SR16 发布

Spring Data 团队宣布发布 Ingalls SR16 维护版本。这个版本获取最新的 Spring Framework 维护版本：4.3.20。Spring Boot 将分别通过其 1.5.17 版本选取 Ingalls SR16 以便于使用。

该服务发布主要包含错误修复和一些依赖性升级，总共修复了 70 多个问题。您可以通过以下链接找到完整的问题列表：

- Ingalls SR16

为了解决这些问题，以下是各个更改日志、文档和工件的链接：

## Ingalls SR16

- Spring Data Commons 1.13.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data JPA 1.11.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Solr 2.1.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data MongoDB 1.10.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data KeyValue 1.2.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Cassandra 1.5.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Gemfire 1.9.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Neo4j 4.2.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data LDAP 1.0.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Envers 1.1.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#)
- Spring Data REST 2.6.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Redis 1.8.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Elasticsearch 2.1.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Couchbase 2.2.16 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)

# Spring Data Ingalls SR15 发布

Spring Data Ingalls SR15 发布，属于维护更新版本。

Ingalls SR15 将会被选入 Spring Boot [1.5.16](#) 中。

这个版本修复了一些 Bug 和升级一些第三方依赖，具体来说，Ingalls SR15 关闭了 [40](#) 问题单。

## Ingalls SR15

- Spring Data Commons 1.13.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data JPA 1.11.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data KeyValue 1.2.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Solr 2.1.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Gemfire 1.9.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Neo4j 4.2.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data MongoDB 1.10.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data for Apache Cassandra 1.5.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data LDAP 1.0.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Envers 1.1.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#)
- Spring Data REST 2.6.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Redis 1.8.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Elasticsearch 2.1.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)
- Spring Data Couchbase 2.2.15 - [Artifacts](#) - [Javadoc](#) - [Documentation](#) - [Changelog](#)

# Spring Security 发布说明

Spring Security 是一个功能强大且可高度自定义的身份验证和访问控制框架。它是保护基于 Spring 的应用程序的事实上的标准。

其特性如下：

- 对身份验证和授权的全面和可扩展的支持
- 防止会话固定，点击劫持，跨站点请求伪造等攻击
- Servlet API 集成
- 支持与 Spring Web MVC 集成

表格 4. Spring Security 2018 年版本发布记录

版本	发布时间	发布说明
5.1.2.RELEASE	2018-11-28 14:44:40	Spring Security 5.1.2 发布
5.0.10.RELEASE	2018-11-28 16:28:30	Spring Security 5.0.10 发布
4.2.10.RELEASE	2018-11-28 18:49:48	Spring Security 4.2.10 发布
4.2.9.RELEASE	2018-10-16 03:23:30	Spring Security 4.2.9 发布
5.1.1.RELEASE	2018-10-15 19:19:20	Spring Security 5.1.1 发布
5.0.9.RELEASE	2018-10-15 19:56:03	Spring Security 5.0.9 发布
5.1.0.RELEASE	18-09-21 13:13:44	Spring Security 5.1.0 发布
4.2.8.RELEASE	2018-09-11 13:26:15	Spring Security 4.2.8 发布
5.0.8.RELEASE	2018-09-10 14:01:43	Spring Security 5.0.8 发布
5.1.0.RC2	2018-09-07 20:56:22	Spring Security 5.1.0.RC2 发布
5.1.0.RC1	2018-08-20 15:36:11	Spring Security 5.1.0.RC1 发布
5.1.0.M2	2018-07-26 20:21:11	
5.0.7.RELEASE	2018-07-26 20:46:31	
5.0.6.RELEASE	2018-06-13 02:23:02	
4.2.7.RELEASE	2018-06-13 02:35:07	
5.1.0.M1	2018-05-15 02:44:09	
5.0.5.RELEASE	2018-05-08 15:22:58	
4.2.6.RELEASE	2018-05-08 17:14:33	
5.0.4.RELEASE	2018-04-04 17:03:55	
4.2.5.RELEASE	2018-03-30 16:34:42	
5.0.3.RELEASE	2018-02-28 13:05:44	
5.0.2.RELEASE	2018-02-20 04:06:06	
4.1.5.RELEASE	2018-01-25 18:32:02	
5.0.1.RELEASE	2018-01-24 21:07:57	
4.2.4.RELEASE	2018-01-24 23:19:43	

# Spring Security 5.1.2 发布

## Spring Security 5.1.2

已经发布，该版本主要是解决部分缺陷、升级第三方依赖以及优化。这个版本将在本周即将发布的 Spring Boot 维护版本中找到。

5.1.2 升级第三方依赖如下：

- Spring Framework 5.1.3
- Spring Data Lovelace-SR3
- Spring Boot 2.1.0
- Reactor Californium-SR3
- Thymeleaf 3.0.11.RELEASE
- GAE 1.9.68

[项目主页](#) | [5.1 文档](#) | [帮助](#)

# Spring Security 5.1.1 发布

## Spring Security 5.1.1

已经发布，该版本主要是解决部分缺陷、升级第三方依赖以及优化。这个版本将在本周即将发布的 Spring Boot 维护版本中找到。

5.1.1 升级第三方依赖如下：

- Spring Framework 5.1.1
- Spring Data Lovelace-SR1
- Spring Boot 2.1.0.M4
- Reactor Californium-SR1
- Thymeleaf 3.0.10.RELEASE
- Jackson 2.9.7
- GAE 1.9.66

[项目主页](#) | [5.1 文档](#) | [帮助](#)

# Spring Security 5.1.0 发布

Spring Security 5.1.0 正式发布，共解决了 50 多个问题单，增加 OAuth  
示例和完善部分文档，升级了依赖：Spring 5.1.0.GA、Spring Boot 2.1.0.M3 和 Reactor Californium-  
RELEASE。

在这里可以了解更多有关 Spring Security 5.1 的新特性。

第三方依赖升级：

- Spring Framework 5.1.0.RELEASE
- Spring Boot 2.1.0.M3
- Spring Data Lovelace
- Reactor Californium Release
- hibernate-validator 6.0.13.Final
- Jetty 9.4.12.v20180830
- jaxb 2.3.0.1
- unboundid-ldapsdk 4.0.8
- htmlunit 2.33
- mockito-core 2.22.0

[项目主页](#) | [参考文档](#) | [问答社区](#)

# Spring Security 5.1.0.RC2 发布

Spring Security 5.1.0.RC2 发布，共解决了 50 多个问题单，其中不少是针对 OAuth2 的配置和自定义能力增强，继续改进 WebFlux 风格的相关安全配置，此外也升级了依赖：Spring 5.1.0.RC3、Spring Boot 2.1.0.M2 和 Reactor Californium-M2。

## 简化OAuth2 DSL

在之前，Spring Security DSL 有两种 OAuth 写法：

```
http
    .oauth2Login()...
```

和

```
http
    .oauth2()
    .client()...
```

这是有道理的，因为一个是身份验证机制，如 `formLogin` 和 `openidLogin`，其他 - 客户端，`resourceServer` 和 `authorizationServer` -- 更像 OAuth 2.0 特有的。

但最后，这种分离感觉就像不必要的额外打字一样，所以我们决定将层次结构对齐，这意味着，从本版本开始，我们现在拥有：

```
http
    .oauth2Login()...
    .oauth2Client()...
    .oauth2ResourceServer()
```

DSL 的重构不会涉及任何功能或特性变更，只需更少的键入。

## WebClient 扩展

开发团队一直在努力使用 WebClient，我们很高兴地宣布一些针对 Servlet 和 WebFlux 应用程序的新 OAuth 2.0 WebClient 扩展。通过这些扩展，可以轻松地在机器之间无缝地传输 OAuth 2.0 权限。

在 [OAuth 2.0 Web Client 相关需求单](#) 中可以看到所有详细信息。

## Servlet 改进

### OAuth2

#### Token 请求配置 (Token Request Configuration)

OAuth 2.0 的第一部分是规范合规性。下一个版本引入了更多支持，可以将 Spring Security

配置为与扩展或偏离规范的提供程序一起使用。

例如，现在可以 [自定义从客户端到授权服务器的令牌请求](#)。

资源服务器声明映射（Resource Server Claims Mapping）

除此之外，Resource Server 还支持 [自定义从传入JWT解析的声明集](#)。当应用程序需要添加或删除声明或需要以自定义方式解析声明时，这很方便：

```
NimbusJwtDecoderJwkSupport decoder = // ...
decoder.setClaimSetConverter(
    MappedJwtClaimSetConverter
        .withDefaults("custom-date", this::convertToInstant));
```

提供更多提供商配置元数据（More Provider Configuration Metadata Available）

规范合规的过程也在继续。在此版本中，添加了对 [从 OIDC Provider Configuration 端点返回的任何元数据](#)的收集和提供的支持。

现在，[ClientRegistration](#) 增加了方法 [getConfigurationMetadata](#)，它与针对特定属性的方法一起，并返回整个提供的属性的映射。

RestOperations 支持

此版本还支持 [对各种端点的 HTTP 请求的完全自定义](#)。

这对于配置超时、发现、缓存以及在与授权服务器通信时充分利用 [RestTemplate](#) 的复杂性非常方便。

其他优化

X.509 Principal 提取

还增加了对 [通过策略派生 X.509 主体](#)的支持。

LDAP 自定义环境变量

添加了配置自定义环境变量的支持，以通知创建 [LdapContext](#)。

WebFlux 改进

OAuth2 资源服务器（Resource Server）

最初为基于 Servlet 的资源服务器发布的几个功能在 WebFlux 端的 RC2 版本中也添加进来了。对响应式特性的支持类似，但有一个很小却很重要的不同点。

在 Spring Security WebFlux 中，每个请求类型具有一个身份验证管理器更为常见。在此版本中，WebFlux 中资源服务器的身份验证管理器配置：

```
http
    .oauth2ResourceServer()
        .authenticationManager(customAuthenticationManager())
```

这在应用程序需要将 **Jwt** 自定义转换为一组授权权限的情况下很有用。

## OAuth2 Client

现在，**@RegisteredOAuth2AuthorizedClient**注解在 WebFlux 中也支持 `client_credentials` 授权模式。

## 重定向跳转 Https

最后但同样重要的是，**https** 重定向支持已添加到 WebFlux，可通过 `http.redirectToHttps()` 或直接通过 `HttpsRedirectWebFilter` 访问。

## 依赖升级

一些依赖升级到最新版本：

- Spring 5.1.0.RC3
- Spring Boot 2.1.0.M2
- hibernate-entitymanager 5.3.6.Final
- nimbus-jose-jwt 6.0.2
- oauth2-oidc-sdk 6.0
- HtmlUnit 2.32.1
- assertj 3.11.1

[项目主页](#) | [参考文档](#) | [问答社区](#)

# Spring Security 5.1.0.RC1 发布

Spring Security 5.1.0.RC1 发布，共解决了 50 多个问题单，其中不少是针对 OAuth2 的配置和自定义能力增强，继续改进 WebFlux 风格的相关安全配置，此外也升级了依赖：Spring 5.1.0.RC2、Spring Boot 2.1.0.M1 和 Reactor Californium-M2。

## Servlet

OAuth2 资源服务器（Resource Server）

Open ID Provider 配置

现在可以通过支持 Open Id Provider Configuration 的任何发行方端点配置 Resource Server：

```
@Bean  
JwtDecoder jwtDecoder() {  
    return JwtDecoders.createDefaultFromIssuer("https://issuer-endpoint");  
}
```

声明验证（Claim Validation）

用户可以通过声明 JwtDecoder bean 来添加自己的验证规则以应用于 Jwt：

```
@Bean  
JwtDecoder jwtDecoder() {  
    String jwkSetUri = "https://issuer-endpoint/.well-known/jwks.json";  
    NimbusJwtDecoderJwkSupport jwtDecoder =  
        new NimbusJwkDecoderJwkSupport(jwkSetUri);  
    OAuth2TokenValidator<Jwt> validator =  
        new DelegatingOAuth2TokenValidator(  
            JwtValidators.createDefault(),  
            new MyCustomValidator());  
    jwtDecoder.setJwtValidator(validator);  
    return jwtDecoder;  
}
```

GrantedAuthority 获取（Extraction）

用户可以自定义从 Jwt 获取 GrantedAuthority 的方式：

```

@Bean
JwtDecoder jwtDecoder() {
    String jwkSetUri = "https://issuer-endpoint/.well-known/jwks.json";
    NimbusJwtDecoderJwkSupport jwtDecoder =
        new NimbusJwkDecoderJwkSupport(jwkSetUri);
    JwtAuthenticationConverter jwtAuthenticationConverter =
        new JwtAuthenticationConverter() {
            protected Collection<GrantedAuthority> extractAuthorities(Jwt jwt) {
                return Arrays.asList(new SimpleGrantedAuthority("app:read"));
            }
        };
    jwtDecoder.setJwtAuthenticationConverter(jwtAuthenticationConverter);
    return jwtDecoder;
}

```

## OAuth2 客户端凭证授予 (OAuth2 Client Credentials Grant)

已添加 [对客户端凭证授予类型](#) 的基本支持。

## Feature-Policy 安全头

基本支持 [Feature-Policy](#):

```

http
.headers()
.featurePolicy("geolocation 'none'");

```

## WebFlux

### OAuth2 资源服务器

基本支持基于响应式实现的 OAuth2 资源服务器 (OAuth2 Resource Servers)。查看示例：[oauth2resourceserver-webflux](#)。

### OAuth2 Login/Client

#### 授权码模式 (Authorization Code Grant)

基本支持基于响应式实现的授权码模式。查看示例：[authcodegrant-webflux](#)。

#### 授权请求解析器 (Authorization Request Resolver)

支持对授权服务器进行身份验证请求的自定义。例如，如果授权服务器需要发送自定义参数，那么使用 [ServerOAuth2AuthorizationRequestResolver](#) 可以很方便处理。它在多租户场景中也很有用，其中请求的元素（如主机名）可能会更改对授权服务器的请求的方式。

## 授权客户端存储库 (Authorized Client Repository)

已添加支持在请求之间自定义授权客户端的持久化实现：

```
http
    .oauth2()
        .client()
            .authorizedClientRepository(new MyCookieBasedClientRepository());
```

## 应用程序安全加固 (Hardening Your Application)

### 安全头 (Secure Headers)

WebFlux 已添加对以下安全标头的支持：

- Content-Security-Policy
- Referrer-Policy
- Feature-Policy

### 跨域资源共享 (CORS)

Webflux 已添加对CORS的支持。

## 依赖升级 (Dependency Updates)

一些依赖升级到最新版本：

- Spring 5.1.0.RC2
- Spring Boot 2.1.0.M1
- Reactor Californium-M2
- hibernate-entitymanager 5.3.5.Final
- hibernate-validator 6.0.12.Final
- unboundid-ldapsdk 4.0.7
- nimbus-jose-jwt 6.0
- oauth2-oidc-sdk 5.64.3
- HtmlUnit 2.32
- assertj 3.11.0

# Spring Security 5.0.10 发布

Spring Security 5.0.10

已经发布，该版本主要是解决部分缺陷、升级第三方依赖以及优化。这个版本将在本周即将发布的 Spring Boot 维护版本中找到。

5.0.10 升级第三方依赖如下：

- Spring Framework 5.0.11
- Spring Data Kay-SR12
- Spring Boot 2.0.6
- Reactor Bismuth-SR14
- Thymeleaf 3.0.11.RELEASE
- Ehcache 2.10.6

[项目主页](#) | [5.0 文档](#) | [帮助](#)

# Spring Security 5.0.9 发布

## Spring Security 5.0.9

已经发布，该版本主要是解决部分缺陷、升级第三方依赖以及优化。这个版本将在本周即将发布的 Spring Boot 维护版本中找到。

5.0.9 升级第三方依赖如下：

- Spring Framework 5.0.10
- Spring Data Kay-SR11
- Reactor Bismuth-SR12
- Thymeleaf 3.0.10.RELEASE
- Jackson 2.9.7
- Unboundid 4.0.8
- JAXB 2.3.0.1

[项目主页](#) | [5.0 文档](#) | [帮助](#)

# Spring Security 5.0.8 发布

Spring Security 5.0.8 发布，该版本主要是解决部分缺陷、升级第三方依赖以及优化。

5.0.8 升级第三方依赖如下：

- Spring Framework 5.0.9
- Spring Data Kay-SR10
- Spring Boot 1.5.15
- Reactor Bismuth-SR11
- Hibernate Validator 6.0.12.Final

[项目主页](#) | [5.0.x 文档](#) | [帮助](#)

# Spring Security 4.2.10 发布

Spring Security 4.2.10

已经发布，该版本主要是解决部分缺陷、升级第三方依赖以及优化。这些版本将在本周即将发布的 Spring Boot 维护版本中找到。

4.2.10 升级第三方依赖如下：

- Spring Framework 4.3.21
- Spring Data Ingalls SR17
- Spring Session 1.3.4
- Jackson 2.8.11.3

[项目主页](#) | [4.x 文档](#) | [帮助](#)

# Spring Security 4.2.9 发布

Spring Security 4.2.9

已经发布，该版本主要是解决部分缺陷、升级第三方依赖以及优化。这个版本将在本周即将发布的 Spring Boot 维护版本中找到。

4.2.9 升级第三方依赖如下：

- Spring Data Ingalls SR16
- Spring Session 1.3.3

[项目主页](#) | [4.x 文档](#) | [帮助](#)

# Spring Security 4.2.8 发布

Spring Security 4.2.8 发布，该版本主要是解决部分缺陷、升级第三方依赖以及优化。

4.2.8 升级第三方依赖如下：

- Spring Framework 4.3.19
- Spring Data Commons 1.13.14
- Spring Boot 1.5.15
- Tomcat 7.0.90

[项目主页](#) | [4.x 文档](#) | [帮助](#)

# Spring Cloud 发布说明

Spring Cloud 为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理、服务发现、断路器、智能路由、微代理、控制总线、一次性令牌、全局锁定、领导选举、分布式会话、集群状态）。分布式系统的协调导致样板（“锅炉板”）模式，使用 Spring Cloud 开发人员可以快速站起来实现这些模式的服务和应用程序。它们适用于任何分布式环境，包括开发人员自己的笔记本电脑、裸机数据中心和 Cloud Foundry 等托管平台。

Spring Cloud 专注于为典型用例提供良好的开箱即用体验，并为其他用户提供可扩展性机制。

- 分布式/版本化配置
- 服务注册和发现
- 路由
- 服务间调用
- 负载均衡
- 断路器
- 全局锁
- 领导选举和状态集群
- 分布式消息

主要项目：

- Spring Cloud Config
- Spring Cloud Netflix
- Spring Cloud Bus
- Spring Cloud Cloudfoundry
- Spring Cloud Open Service Broker
- Spring Cloud Cluster
- Spring Cloud Consul
- Spring Cloud Security
- Spring Cloud Sleuth
- Spring Cloud Data Flow
- Spring Cloud Stream
- Spring Cloud Task
- Spring Cloud Zookeeper
- Spring Cloud AWS
- Spring Cloud Connectors

- Spring Cloud CLI
- Spring Cloud Contract
- Spring Cloud Gateway
- Spring Cloud OpenFeign
- Spring Cloud Pipelines
- Spring Cloud Function

表格 5. Spring Cloud 2018 年版本发布记录

版本	发布时间	发布说明
Greenwich.RC2	2018-12-21 01:06:03	Spring Cloud Greenwich.RC2 发布
Greenwich.RC1	2018-12-12 00:55:37	Spring Cloud Greenwich.RC1 发布
Greenwich.M3	2018-11-19 23:18:46	Spring Cloud Greenwich.M3 发布
Greenwich.M2	2018-11-18 10:26:25	
Finchley.SR2	2018-10-23 20:01:01	Spring Cloud Finchley.SR2 发布
Edgware.SR5	2018-10-16 14:26:35	Spring Cloud Edgware.SR5 发布
Greenwich.M1	2018-09-24 01:27:10	
Finchley.SR1	2018-07-31 20:45:31	
Edgware.SR4	2018-06-30 00:01:58	
Finchley.RELEASE	2018-06-19 01:57:34	
Finchley.RC2	2018-05-29 13:44:47	
Finchley.RC1	2018-04-20 18:25:43	
Edgware.SR3	2018-03-27 10:36:19	
Finchley.M9	2018-03-22 19:09:15	
Finchley.M8	2018-03-02 19:07:53	
Finchley.M7	2018-02-27 16:20:28	
Finchley.M6	2018-02-10 03:41:08	
Edgware.SR2	2018-02-09 18:47:06	
Edgware.SR1	2018-01-16 18:15:19	
Finchley.M5	2018-01-02 13:25:18	

# Spring Cloud Greenwich.RC2 发布

Ryan Baxter 在博客宣布了 Spring Cloud Greenwich.RC2 发布。

Spring Cloud Greenwich.RC2 可以在 Spring Milestone 库获取到。也可以查看 Greenwich 发布说明了解更多信息。

## Greenwich 版本值得注意的更新

### Spring Cloud Contract

添加了二进制载荷的支持 (GH-818)。

### Spring Cloud Security

Spring Cloud Gateway 过滤器新增对 OAuth2 的支持 (GH-141)。一个示例演示应用程序在 [这里](#)。

### Spring Cloud Kubernetes

添加了一个模块来检测 Istio 的存在 (GH-233)。

### Spring Cloud Openfeign

升级 OpenFeign 10.1.0 (GH-85)。

### Spring Cloud Task

详见 [博客链接](#)。

### Spring Cloud Config

添加了 EnvironmentRepository 以支持 CredHub 后端 (GH-1211)。

以下模块作为 Greenwich.RC2 的一部分进行了更新：

模块	版本	问题
Spring Cloud Security	2.1.0.RC3	(问题)
Spring Cloud Vault	2.1.0.RC1	
Spring Cloud Contract	2.1.0.RC3	(问题)
Spring Cloud Kubernetes	1.0.0.RC2	(问题)
Spring Cloud Commons	2.1.0.RC2	
Spring Cloud Zookeeper	2.1.0.RC2	
Spring Cloud Openfeign	2.1.0.RC3	(问题)
Spring Cloud Aws	2.1.0.RC1	
Spring Cloud Starter	Greenwich.RC2	
Spring Cloud Bus	2.1.0.RC3	

模块	版本	问题
Spring Cloud Sleuth	2.1.0.RC3	(问题)
Spring Cloud Netflix	2.1.0.RC3	(问题)
Spring Cloud Stream	Fishtown.RC4	(问题)
Spring Cloud Gcp	1.1.0.RC2	
Spring Cloud Cloudfoundry	2.1.0.RC2	
Spring Cloud Build	2.0.0.RC3	(问题)
Spring Cloud Dependencies	Greenwich.RC2	
Spring Cloud Task	2.1.0.M2	
Spring Cloud Release	Greenwich.RC2	
Spring Cloud Function	2.0.0.RC3	(问题)
Spring Cloud	Greenwich.RC2	
Spring Cloud Consul	2.1.0.RC3	(问题)
Spring Cloud Config	2.1.0.RC3	(问题)
Spring Cloud Gateway	2.1.0.RC3	(问题)

要使用BOM开始使用Maven（仅限依赖关系管理）：

```
<repositories>
  <repository>
    <id>spring-milestones</id>
    <name>Spring Milestones</name>
    <url>http://repo.spring.io/milestone</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Greenwich.RC2</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>
  ...
</dependencies>
```

或者，Gradle：

```

buildscript {
    dependencies {
        classpath "io.spring.gradle:dependency-management-plugin:1.0.6.RELEASE"
    }
}

repositories {
    maven {
        url 'http://repo.spring.io/milestone'
    }
}

apply plugin: "io.spring.dependency-management"

dependencyManagement {
    imports {
        mavenBom 'org.springframework.cloud:spring-cloud-dependencies:Greenwich.RC2'
    }
}

dependencies {
    compile 'org.springframework.cloud:spring-cloud-starter-config'
    compile 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    ...
}

```

[项目主页](#) | [Github](#) | [Gitter](#) | [Stackoverflow](#) | [Twitter](#)

NOTE

原文: <https://spring.io/blog/2018/12/21/spring-cloud-greenwich-rc2-is-now-available>  
作者: Ryan Baxter  
编译: 春之雨

# Spring Cloud Greenwich.RC1 发布

Spencer Gibb 在博客宣布了 Spring Cloud Greenwich.RC1 发布。

Spring Cloud Greenwich.RC1 可以在 Spring Milestone 库获取到。也可以查看 Greenwich 发布说明了解更多信息。

## 寿命终止 (EOL) 提醒

Dalston 发布版本将于2018年底进入 EOL 状态。

## Greenwich 版本值得注意的更新

此里程碑与 Spring Boot 2.1.1.RELEASE 兼容。已对 Java 11 兼容性进行了更改。有关更多信息，请参阅 Github 项目。

## Spring Cloud Netflix 项目进入维护模式

最近，Netflix 宣布 Hystrix 正在进入维护模式。自 2016 年以来，Ribbon 也已处于类似状态。虽然 Hystrix 和 Ribbon 现在处于维护模式，但它们仍然在 Netflix 上大规模部署。

Hystrix Dashboard 和 Turbine 已被 Atlas 取代。这些项目的最后提交分别是 2 年和 4 年前。Zuul 1 和 Archaius 1 都被后来不兼容的版本所取代。

以下 Spring Cloud Netflix 模块和相应的起步依赖将进入维护模式：

1. spring-cloud-netflix-archaius
2. spring-cloud-netflix-hystrix-contract
3. spring-cloud-netflix-hystrix-dashboard
4. spring-cloud-netflix-hystrix-stream
5. spring-cloud-netflix-hystrix
6. spring-cloud-netflix-ribbon
7. spring-cloud-netflix-turbine-stream
8. spring-cloud-netflix-turbine
9. spring-cloud-netflix-zuul

这不包括 Eureka 或并发限制模块。

## 什么是维护模式？

将模块置于维护模式意味着 Spring Cloud 团队将不再向模块添加新功能。我们将修复重大程序错误和安全问题，我们还将考虑并审查社区的小的拉取请求 (PR)。

我们打算继续支持这些模块，从 Greenwich 正式发布可用之后算起至少一年的时间。

## 替代项目

我们建议将以下内容替换为这些模块提供的功能。

当前	替换
Hystrix	Resilience4j
Hystrix Dashboard / Turbine	Micrometer + Monitoring System
Ribbon	Spring Cloud Loadbalancer
Zuul 1	Spring Cloud Gateway
Archaius 1	Spring Boot external config + Spring Cloud Config

查看有关 Spring Cloud Loadbalancer 的后续博客文章，并与新的 Netflix 项目 Concurrency Limits 集成。

### Spring Cloud Kubernetes

在引导期间使用 [KubernetesDiscoveryClient](#) 进行了增强以及许多文档更新。

### Spring Cloud Commons

对如何加载 BootstrapConfiguration 类以适应 Java 11 中的行为进行了更改。

### Spring Cloud Contract

修复 Bug。

### Spring Cloud Openfeign

支持增加注解 [@QueryMap](#)。

### Spring Cloud Zookeeper

支持增加字段 [ServiceInstance.instanceId](#)。

### Spring Cloud Stream

参见 Fishtown.RC2 的：[文章](#)

### Spring Cloud Sleuth

支持 gRPC 插桩检测。

### Spring Cloud Bus

修复了一个当使用最新版本的 Spring Cloud Stream 时会出现 Bus 无法正常运行的严重 BUG。

### Spring Cloud Gcp

文档更新和错误修复。

### Spring Cloud Function

支持 Kotlin lambdas，以及其他改进和 Bug 修复。

## Spring Cloud Consul

支持增加 `ServiceInstance.instanceId` 属性，以及 Bug 修复。

## Spring Cloud Config

修复 webhooks 和 Spring Cloud Bus 相关问题。

## Spring Cloud Gateway

在各自的路由谓词中支持添加多个路径和主机，并可自定义在某些情况下返回的 HTTP 状态码，以及错误修复。

以下模块作为 `Greenwich.RC1` 的一部分进行了更新：

模块	版本	问题
Spring Cloud Kubernetes	1.0.0.RC1	( <a href="#">问题</a> )
Spring Cloud Commons	2.1.0.RC2	( <a href="#">问题</a> )
Spring Cloud Contract	2.1.0.RC2	( <a href="#">问题</a> )
Spring Cloud Openfeign	2.1.0.RC2	( <a href="#">问题</a> )
Spring Cloud Zookeeper	2.1.0.RC2	( <a href="#">问题</a> )
Spring Cloud Stream	Fishtown.RC2	( <a href="#">文章</a> )
Spring Cloud Sleuth	2.1.0.RC2	( <a href="#">问题</a> )
Spring Cloud Bus	2.1.0.RC2	( <a href="#">问题</a> )
Spring Cloud Netflix	2.1.0.RC2	( <a href="#">问题</a> )
Spring Cloud Gcp	1.1.0.RC1	( <a href="#">问题</a> )
Spring Cloud Cloudfoundry	2.1.0.RC2	
Spring Cloud Function	2.0.0.RC2	( <a href="#">问题</a> )
Spring Cloud Task	2.1.0.M1	( <a href="#">问题</a> )
Spring Cloud Consul	2.1.0.RC2	( <a href="#">问题</a> )
Spring Cloud Config	2.1.0.RC2	( <a href="#">问题</a> )
Spring Cloud Gateway	2.1.0.RC2	( <a href="#">问题</a> )

要使用BOM开始使用Maven（仅限依赖关系管理）：

```
<repositories>
  <repository>
    <id>spring-milestones</id>
    <name>Spring Milestones</name>
    <url>http://repo.spring.io/milestone</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Greenwich.RC1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>
  ...
</dependencies>
```

或者，Gradle：

```
buildscript {
    dependencies {
        classpath "io.spring.gradle:dependency-management-plugin:1.0.6.RELEASE"
    }
}

repositories {
    maven {
        url 'http://repo.spring.io/milestone'
    }
}

apply plugin: "io.spring.dependency-management"

dependencyManagement {
    imports {
        mavenBom 'org.springframework.cloud:spring-cloud-dependencies:Greenwich.RC1'
    }
}

dependencies {
    compile 'org.springframework.cloud:spring-cloud-starter-config'
    compile 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    ...
}
```

[项目主页](#) | [Github](#) | [Gitter](#) | [Stackoverflow](#) | [Twitter](#)

NOTE 原文: <https://spring.io/blog/2018/12/12/spring-cloud-greenwich-rc1-available-now>  
作者: Spencer Gibb  
编译: 春之雨

# Spring Cloud Greenwich.M3 发布

Spencer Gibb 在博客宣布了 Spring Cloud Greenwich.M3 发布。

Spring Cloud Greenwich.M3 可以在 Spring Milestone 仓库获取到。也可以查看 Greenwich 发布说明了解更多信息。

## Greenwich 版本值得注意的更新

此里程碑与 Spring Boot 2.1.0.RELEASE 兼容。已对 Java 11 兼容性进行了更改。有关更多信息，请参阅 Github 项目。

### Spring Cloud Bus

修复了 Spring Cloud Stream 的兼容性问题。

### Spring Cloud Commons

将 `instanceld` 添加到 `ServiceInstance` 接口。还添加了 `ReactiveLoadBalancer` 接口并提供了 Reactor 的实现。

### Spring Cloud Config

一些错误修复和小的功能增强。

### Spring Cloud Contract

添加对 `WebTestClient` 和 JUnit 5 扩展的支持

### Spring Cloud Consul

Bug 修复。

### Spring Cloud Function

小的功能增强和错误修复。

### Spring Cloud Gateway

添加了重写响应头过滤器和错误修复。

### Spring Cloud Gcp

小的功能增强和错误修复。

### Spring Cloud Kubernetes

现在可以将 `ServiceInstance` 元数据配置为来自 Kubernetes Labels, Annotations 和 Ports。其他一些小问题和错误修复。

### Spring Cloud Netflix

一些依赖项版本升级。

## Spring Cloud Openfeign

添加缺少的配置元数据。

## Spring Cloud Sleuth

对 Reactor 和 Reactor Netty 插桩的各种改进以及其他微小改进。

## Spring Cloud Stream

小的功能增强和错误修复。

## Spring Cloud Task

现在通过自动配置启用。

## Spring Cloud Vault

添加对 Azure 和 GCP 身份验证的支持以及一些依赖项更新。

以下模块作为 [Greenwich.M3](#) 的一部分进行了更新：

模块	版本	问题
Spring Cloud	Greenwich.M3	
Spring Cloud Aws	2.0.1.RELEASE	( <a href="#">问题</a> )
Spring Cloud Bus	2.1.0.M2	( <a href="#">问题</a> )
Spring Cloud Cloudfoundry	2.1.0.M1	
Spring Cloud Commons	2.1.0.M2	( <a href="#">问题</a> )
Spring Cloud Config	2.1.0.M3	( <a href="#">问题</a> )
Spring Cloud Contract	2.1.0.M2	( <a href="#">问题</a> )
Spring Cloud Consul	2.1.0.M2	( <a href="#">问题</a> )
Spring Cloud Function	2.0.0.RC2	( <a href="#">问题</a> )
Spring Cloud Gateway	2.1.0.M3	( <a href="#">问题</a> )
Spring Cloud Gcp	1.1.0.M3	
Spring Cloud Kubernetes	1.0.0.M2	( <a href="#">问题</a> )
Spring Cloud Netflix	2.1.0.M3	( <a href="#">问题</a> )
Spring Cloud Openfeign	2.1.0.M2	
Spring Cloud Security	2.1.0.M1	
Spring Cloud Sleuth	2.1.0.M2	( <a href="#">问题</a> )
Spring Cloud Stream	Fishtown.RC2	<a href="#">文章</a>
Spring Cloud Task	2.1.0.M1	( <a href="#">问题</a> )

模块	版本	问题
Spring Cloud Vault	2.1.0.M2	(问题)
Spring Cloud Zookeeper	2.1.0.M1	

要使用BOM开始使用Maven（仅限依赖关系管理）：

```

<repositories>
  <repository>
    <id>spring-milestones</id>
    <name>Spring Milestones</name>
    <url>http://repo.spring.io/milestone</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Greenwich.M3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>
  ...
</dependencies>

```

或者，Gradle：

```
buildscript {
    dependencies {
        classpath "io.spring.gradle:dependency-management-plugin:1.0.2.RELEASE"
    }
}

repositories {
    maven {
        url 'http://repo.spring.io/milestone'
    }
}

apply plugin: "io.spring.dependency-management"

dependencyManagement {
    imports {
        mavenBom 'org.springframework.cloud:spring-cloud-dependencies:Greenwich.M3'
    }
}

dependencies {
    compile 'org.springframework.cloud:spring-cloud-starter-config'
    compile 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    ...
}
```

[项目主页](#) | [Github](#) | [Gitter](#) | [Stackoverflow](#) | [Twitter](#)

NOTE 原文：<https://spring.io/blog/2018/11/21/spring-cloud-greenwich-m3-is-available>  
作者：Spencer Gibb  
编译：春之雨

# Spring Cloud Finchley.SR2 发布

Spring Cloud Finchley SR2 (Service Release) 已经发布，可以在[Maven Central](#)上获取到。

此次更新，主要是升级到Spring Boot [2.0.6.RELEASE](#)，其中包含的几个模块也都是解决一些缺陷和依赖升级，所以建议升级到此版本。您可以查看 Finchley [发布说明](#) 以获取更多信息。

## 模块更新说明

具体而言，下面是每个更新模块的详细介绍。

### Spring Cloud Gateway

- 支持Hystrix超时中的WebFlux错误处理 [#553](#)
- 在PEM文件中读取多个证书 [#583](#)
- 支持配置TLS超时 [#578](#)
- [ModifyRequestBodyGatewayFilterFactory](#)适当更新请求头 [#492](#)
- 支持重定向过滤器中的相对路径跳转 [#468](#)
- 其他缺陷修复

### Spring Cloud Sleuth

- 缺陷修复

### Spring Cloud Config

- 传播 Git 克隆错误 [#1160](#)
- Gitee 和 Gogs 的 webhook 支持 [#1140](#)
- 缺陷修复

### Spring Cloud Netflix

- 支持接受Sidecars中的所有证书 [#3224](#)
- [HystrixCommands.fallback](#) 现在支持接受导致回调被调用的 [Throwable](#) [#3210](#)
- 缺陷修复

### Spring Cloud Commons

- 使用 [@LoadBalanced WebClient](#) 时更好的错误处理 [#386](#)
- [InstancePreRegisteredEvent](#) 在[服务注册](#)之前触发
- 缺陷修复

### Spring Cloud Contract

- 缺陷修复

## Spring Cloud Vault

- 缺陷修复

## Spring Cloud Openfeign

- 在Feign客户端中添加对 `MultipartFile` 的支持 #62
- 缺陷修复

## Spring Cloud Security

- 依赖升级

## Spring Cloud AWS

- 缺陷修复

## 模块及版本

以下模块作为Finchley.SR2的一部分进行了更新：

表格 6. 更新的模块及版本

模块	版本
Spring Cloud Gateway	2.0.2.RELEASE
Spring Cloud Sleuth	2.0.2.RELEASE
Spring Cloud Config	2.0.2.RELEASE
Spring Cloud Netflix	2.0.2.RELEASE
Spring Cloud Commons	2.0.2.RELEASE
Spring Cloud Contract	2.0.2.RELEASE
Spring Cloud Vault	2.0.2.RELEASE
Spring Cloud Openfeign	2.0.2.RELEASE
Spring Cloud AWS	2.0.1.RELEASE
Spring Cloud Cloud Foundry	2.0.1.RELEASE
Spring Cloud Security	2.0.1.RELEASE

## 如何使用

要使用BOM开始使用Maven（仅限依赖关系管理）：

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Finchley.SR2</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>
  ...
</dependencies>
```

或者，使用Gradle：

```
buildscript {
    dependencies {
        classpath "io.spring.gradle:dependency-management-plugin:1.0.5.RELEASE"
    }
}

apply plugin: "io.spring.dependency-management"

dependencyManagement {
    imports {
        mavenBom 'org.springframework.cloud:spring-cloud-dependencies:Finchley.SR2'
    }
}

dependencies {
    compile 'org.springframework.cloud:spring-cloud-starter-config'
    compile 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    ...
}
```

一如既往，我们非常欢迎来自[GitHub](#), [Gitter](#), [Stack Overflow](#) 以及[Twitter](#)上的反馈。

# Spring Cloud Edgware.SR5 发布

Spring Cloud Edgware Service Release 5 (SR5) 现已发布，可以从 [repo.spring.io](#) 和 Maven Central 获取到。

## 值得注意的更新

### Spring Cloud Commons

- 缺陷修复

### Spring Cloud Config

- 文档和缺陷修复

### Spring Cloud Contract

- 缺陷修复

通过 #707，我们有一个测试监听器来处理关闭和启动WireMock服务器的问题。由于此更改，如果要为测试桩重用相同的端口，则不再需要在测试中设置 `@DirtiesContext`。

### Spring Cloud Sleuth

- 缺陷修复

通过 #1077，我们增加了对单个请求头B3传播的支持。

### Spring Cloud Netflix

缺陷修复

### Spring Cloud Vault

缺陷修复

### Spring Cloud Consul

缺陷修复

以下模块作为 **Edgware.SR5** 的一部分进行了更新：

模块	版本
Spring Cloud AWS	1.2.3.RELEASE
Spring Cloud Contract	1.2.6.RELEASE
Spring Cloud Consul	1.3.5.RELEASE
Spring Cloud Zookeeper	1.2.2.RELEASE
Spring Cloud Sleuth	1.3.5.RELEASE
Spring Cloud Config	1.4.5.RELEASE

模块	版本
Spring Cloud Netflix	1.4.6.RELEASE
Spring Cloud Commons	1.3.5.RELEASE
Spring Cloud Bus	1.3.4.RELEASE
Spring Cloud Security	1.2.3.RELEASE
Spring Cloud Cloudfoundry	1.1.2.RELEASE
Spring Cloud Function	1.0.1.RELEASE
Spring Cloud Vault	1.1.2.RELEASE
Spring Cloud Gateway	1.0.2.RELEASE

要使用BOM开始使用Maven（仅限依赖关系管理）：

```

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Edgware.SR5</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>
  ...
</dependencies>

```

或者，Gradle：

```
buildscript {  
    dependencies {  
        classpath "io.spring.gradle:dependency-management-plugin:1.0.2.RELEASE"  
    }  
}  
  
apply plugin: "io.spring.dependency-management"  
  
dependencyManagement {  
    imports {  
        mavenBom 'org.springframework.cloud:spring-cloud-dependencies:Edgware.SR5'  
    }  
}  
  
dependencies {  
    compile 'org.springframework.cloud:spring-cloud-starter-config'  
    compile 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'  
    ...  
}
```

NOTE

原文：<https://spring.io/blog/2018/10/17/spring-cloud-edgware-sr5-has-been-released>  
作者：Ryan Baxter  
翻译：春之雨

# Spring Cloud for Alibaba 0.2.1 发布

NOTE 以下内容由来自阿里巴巴的阿里中间件团队提供，并由 Spencer Gibb 发布。

大家好，很高兴地告诉大家，今天 Spring Cloud Alibaba 0.2.1.RELEASE 版本发布了。

本次发布版本主要亮点：

- Spring Cloud Alibaba 新增两个模块：`spring-cloud-alibaba-schedulerx` 和 `spring-cloud-stream-binder-rocketmq`。
- 增加新特性：`spring-cloud-alibaba-nacos` 和 `spring-cloud-alibaba-sentinel`。
- 修复上一个版本的 BUG

**RocketMQ** Apache RocketMQ™ 是基于 Java 的高性能、高吞吐量的分布式消息和流计算平台。`spring-cloud-stream-binder-rocketmq` 模块基于 `Spring Integration` 和 `Spring Cloud Stream`。使得开发者在使用 Spring Cloud Stream 和 Spring Cloud Bus 时候可以选择使用 RocketMQ 作为消息中间件。

## AliCloud Schedulerx

阿里中间件团队开发的一款分布式任务调度产品，支持周期性的任务与固定时间点触发任务。

## Spring Cloud Alibaba

项目由两部分组成：包含了阿里巴巴的开源组件和阿里云的产品。旨在利用众所周知的 Spring 框架的设计模式和抽象能力，以便为 Java 开发人员使用阿里巴巴产品带来 Spring Boot 和 Spring Cloud 的优势。

Spring Cloud Alibaba 项目是由阿里巴巴维护的社区项目。

NOTE

**注意：**版本 0.2.1.RELEASE 对应的是 Spring Cloud Finchley 版本，版本 0.1.1.RELEASE 对应的是 Spring Cloud Edgware 版本。0.1.1.RELEASE 同样包括 0.2.1 中相应的组件和功能。

## 模块介绍

### Spring Cloud Stream Binder RocketMQ

- 实现 Spring Cloud Stream API 的消息抽象。
- 包括对交易消息的支持。
- 包括支持在消费者端过滤带有标签和 SQL 表达式的消息，以及有序、并发和广播消息消费。

### Spring Cloud Alibaba Cloud SchedulerX

- 提供精准，高度可靠且高度可用的预定作业调度服务，响应时间在几秒钟内。
- 支持丰富的作业执行模型，包括独立执行，广播执行和分布式子作业执行。

### Spring Cloud Alibaba Nacos Config

- 升级 Nacos 客户端版本到 0.6.2
- 支持从多个 dataid 和 groupid 获取和侦听配置，以及根据 dataid 和 groupid 的组合设置优先级

### Spring Cloud Alibaba Nacos Discovery

- 升级 Nacos 客户端版本到 0.6.2

- 支持在 Nacos 控制台上将服务实例设置为不可用状态，以便在服务发现期间自动过滤掉服务
- 支持在初始化服务发现时忽略本地缓存

## Spring Cloud Alibaba Sentinel

- 添加了对 Feign 的支持，以便它现在与所有 `@FeignClient` 属性兼容，包括 `fallback` 和 `fallbackFactory`。
- 添加了对参数流控制和集群流控制的支持。
- 重构 `ReadableDataSource` 的设计，以提供更加用户友好的方式来配置持久性 Sentinel 规则。
- 当 `RestTemplate` 降级后优化了 Sentinel 的后处理机制。
- 添加了与 Sentinel 配置信息对应的一些属性，例如日志目录和日志文件名。

## 如何使用

这些组件在 Spring release 仓库中，可以通过如下 BOM 来使用：

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-alibaba-dependencies</artifactId>
      <version>0.2.1.RELEASE</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

## 后续计划

### Spring Cloud Alibaba Cloud SLS

针对日志类数据的一站式服务，在阿里巴巴集团经历大量大数据场景锤炼而成。您无需开发就能快捷完成日志数据采集、消费、投递以及查询分析等功能，提升运维、运营效率，建立数据技术（DT）时代海量日志处理能力。

**Spring Cloud Dubbo** Dubbo 是一个流行的开源 RPC 框架。我们将把 Dubbo 整合到 Spring Cloud Alibaba 中，以便您在开发 Dubbo 应用程序时享受 Spring Cloud 的便利。

[项目主页](#) | [GitHub](#) | [问题跟踪](#) | [中文文档](#)

NOTE

原文：<https://spring.io/blog/2018/12/21/spring-cloud-for-alibaba-0-2-1-released>  
作者：Alibaba Aliware Team

# Spring Cloud for Alibaba 0.2.0 发布

大家好，很高兴地告诉大家，今天 Spring Cloud Alibaba 的第一个版本发布了。

## Spring Cloud

Alibaba项目由两部分组成：包含了阿里巴巴的开源组件和阿里云的产品。旨在利用众所周知的 Spring 框架的设计模式和抽象能力，以便为Java开发人员使用阿里巴巴产品带来Spring Boot和Spring Cloud的优势。

Spring Cloud Alibaba 项目是由阿里巴巴维护的社区项目。

### NOTE

**注意：** 版本 0.2.0.RELEASE 对应的是 Spring Boot 2.x 版本，版本 0.1.0.RELEASE 对应的是 Spring Boot 1.x 版本。

## 阿里巴巴开源组件

其中阿里巴巴开源组件的命名前缀为 **spring-cloud-alibaba**，提供了如下特性：

### 服务发现 (Service Discovery)

**spring-cloud-alibaba-nacos-discovery-starter** 实现了 spring cloud common 中定义的 registry 相关规范接口 **NacosAutoServiceRegistration**, **NacosServiceRegistry** 和 **NacosDiscoveryClient** 等，引入依赖并添加一些简单的配置即可将你的服务注册到 Nacos Server 中，并且支持与 Ribbon 的集成。

### 配置管理 (Configuration Management)

**spring-cloud-alibaba-nacos-config-starter** 中 **NacosPropertySourceLocator** 实现了 **PropertySourceLocator** 接口，引入依赖并添加一些简单的配置即可从 Nacos Server 中获取应用配置并设置在 Spring 的 Environment 中。而且无需依赖其他组件即可支持配置的实时推送和推送状态查询。

### 高可用防护

**spring-cloud-alibaba-sentinel-starter** 默认集成了 Servlet、RestTemplate、Dubbo、RocketMQ 的限流(Flow Control)降级(Circuit Breaking and Concurrency)，只需要引入依赖即可完成限流降级的集成动作。并支持在应用运行状态下通过 Sentinel 控制台来实时修改限流降级的策略和阈值。

## 阿里云产品组件

阿里云的产品组件的命名前缀为 **spring-cloud-alicloud**，提供了如下特性：

### 服务发现服务 (Application Naming Service)

阿里云提供的服务发现服务ANS，除了服务发现的基本功能外，提供了更低成本SaaS 化服务发现服务，同时在接口的调用中加入了加密逻辑，更好地保护你的服务。

### 配置管理服务 (Application Configuration Management)

阿里云提供的 Nacos 配置管理服务ACM，除了服务发现的基本功能外，提供了更低成本的 SaaS 化配置管理服务，以及更加安全的配置管理，并且还包含了完整的推送轨迹查询。

### 对象存储服务 (Object Storage Service)

阿里云提供的海量、安全、低成本、高可靠的云存储服务OSS。支持在任何应用、任何时间、任何地点存储和访问任意类型的数据。只需要自动注入一个 OSS Client，即可直接使用存储与下载功能。

## 如何使用 (How to Use)

这些组件在 Spring release 仓库中，可以通过如下 BOM 来使用：

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-alibaba-dependencies</artifactId>
      <version>0.2.0.RELEASE</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

## 后续计划 (What Next?)

**RocketMQ** Apache RocketMQ™ 是基于 Java 的高性能、高吞吐量的分布式消息和流计算平台。**spring-cloud-stream-binder-rocket** 模块基于 **Spring Integration** 和 **Spring Cloud Stream**，使得开发者在使用 Spring Cloud Stream 和 Spring Cloud Bus 时候可以选择使用 RocketMQ 作为消息中间件。

### AliCloud Schedulerx

阿里中间件团队开发的一款分布式任务调度产品，支持周期性的任务与固定时间点触发任务。

### AliCloud SLS

针对日志类数据的一站式服务，在阿里巴巴集团经历大量大数据场景锤炼而成。您无需开发就能快捷完成日志数据采集、消费、投递以及查询分析等功能，提升运维、运营效率，建立数据技术 (DT) 时代海量日志处理能力。

[项目主页](#) | [GitHub](#) | [问题跟踪](#) | [中文文档](#)

NOTE

原文：<https://spring.io/blog/2018/10/30/spring-cloud-for-alibaba-0-2-0-released>  
作者：Spencer Gibb、肖京

# Spring Cloud Data Flow 1.7 正式发布

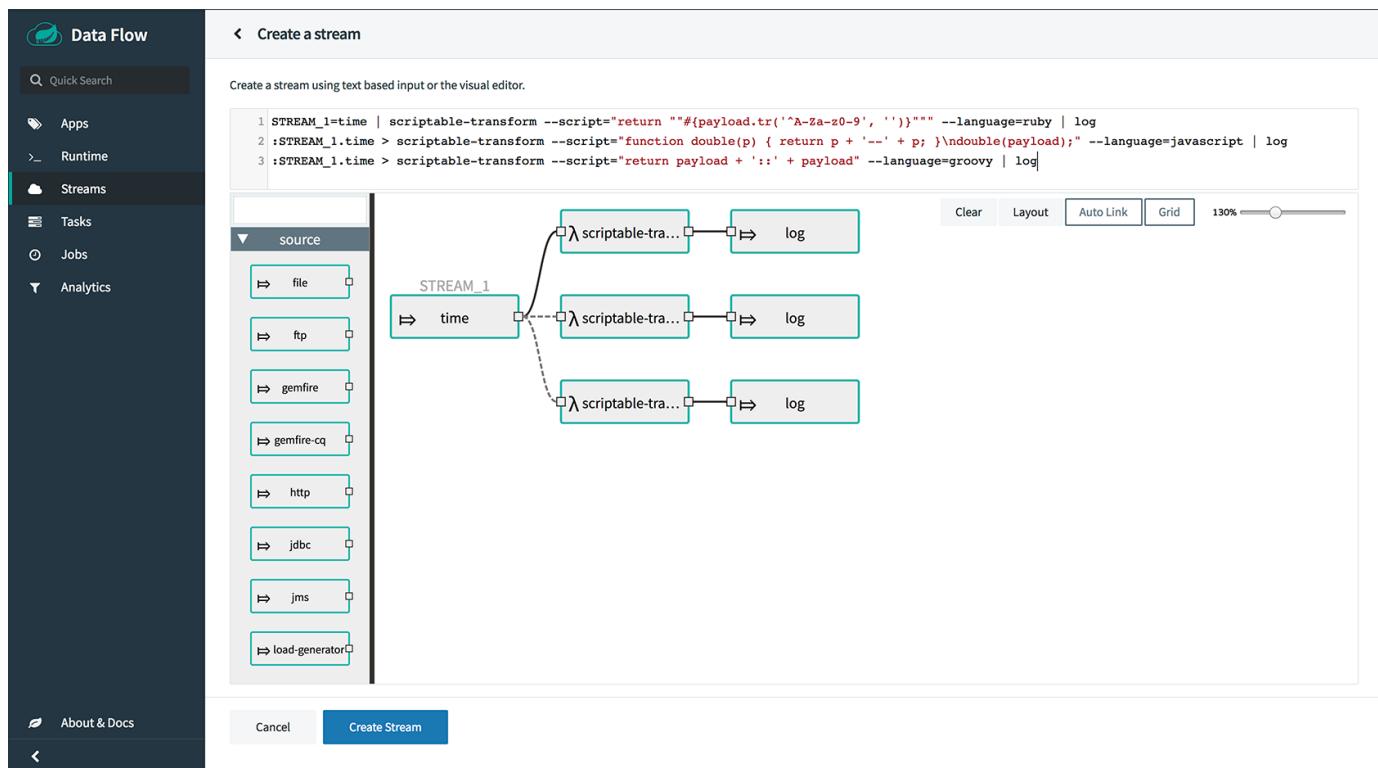
Spring Cloud Data Flow 团队宣布 Spring Cloud Data Flow 1.7.0 发布。请按照 Local Server, Cloud Foundry 和 Kubernetes 的入门指南进行操作。

## 以下是重点

- 改进的 UI
- 流应用程序 DSL
- 审计跟踪
- 并发任务启动限制
- 流和任务校验
- 强制升级流应用
- 支持在 Kubernetes 上调度任务

## 改进 UI

UI 具有全新的外观。导航已从标签换成了左侧导航系统。这为使用 Flo 设计师创建流提供了更多的屏幕空间，通过最小化左侧导航可以获得更多的屏幕空间。有一种快速搜索功能，可以搜索所有不同的数据流 (Data Flow) 类别。添加了其他颜色和整体主题更改，使 UI 看起来更生动。更深入的核心，路由管理得到了改进，我们使用 BrowserStack / SauceLabs 增加了端到端的测试覆盖率。如果白名单为空，则属性白名单功能已经过优化，默认情况下不会显示所有应用程序属性。



## 流应用程序DSL

并非所有用例都可以通过线性管道 (pipeline) 来解决，其中数据从源到处理器流到接收器，并且每个应用程序连接一个目的地。某些用例需要一组具有多个输入和输出的应用程序。Spring Cloud Stream

通过使用用户定义的绑定接口支持此拓扑，但数据流不支持此拓扑。在 Kafka Streams 应用程序中有多个输入和输出也很常见。

此外，并非所有用例都使用 Spring Cloud Stream 应用程序解决。可以仅使用 Spring Integration 编写向 Kafka 或 RabbitMQ 应用程序发送同步请求/回复消息的 http 网关应用程序。

在这些情况下，Data Flow 无法判定从一个应用程序到另一个应用程序的数据流，因此无法像使用 Stream Pipeline DSL 时那样设置应用程序的目标属性。

为了解决这些使用场景，我们引入了 Stream Application DSL。此 DSL 使用，（逗号）而不是 |（管道符号）来指示数据流不应配置应用程序的绑定属性。相反，开发人员需要设置适当的部署属性以“连接”应用程序。下面使用 DSL 来演示 EIP Cafe 示例：

```
dataflow:> stream create --definition "orderGeneratorApp, baristaApp, hotDrinkDeliveryApp, coldDrinkDeliveryApp" --name myCafeStream
```

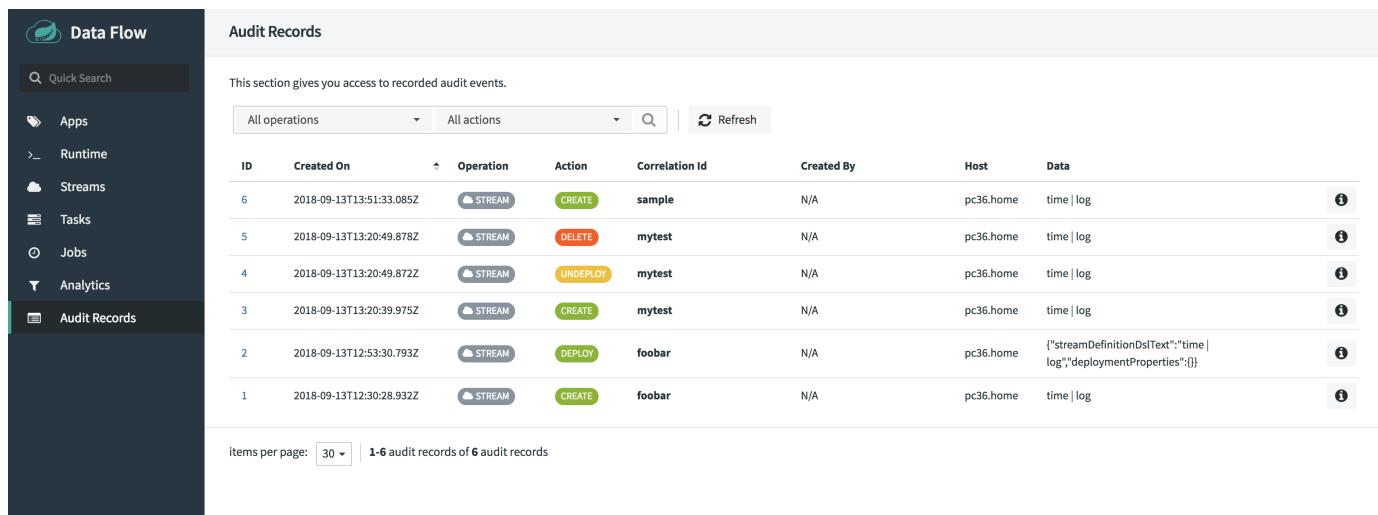
DSL 中列出的应用程序需要注册为 **--type app**。

在这个流中，**baristaApp** 有两个输出目标，分别由 **hotDrinkDeliveryApp** 和 **coldDrinkDeliveryApp** 使用。部署流时，设置目标属性，使目标与所需的数据流匹配，例如：

```
app.baristaApp.spring.cloud.stream.bindings.hotDrinks.destination=hotDrinksDest
app.baristaApp.spring.cloud.stream.bindings.coldDrinks.destination=coldDrinksDest
app.hotDrinkDeliveryApp.spring.cloud.stream.bindings.input.destination=hotDrinksDest
app.coldDrinkDeliveryApp.spring.cloud.stream.bindings.input.destination=coldDrinksDest
```

## 审计跟踪

为了帮助回答“谁做了什么，何时做什么？”的问题。引入了审计跟踪来存储涉及应用程序注册、计划、流和任务的操作。对于应用程序和计划，将审核创建和删除操作。对于流，将审核创建、删除、部署、取消部署、更新和回滚。对于任务，审计创建，启动和破坏。审计信息在 UI 中可用于查询。访问 shell 中的审计信息也将提供。



The screenshot shows the Data Flow UI with the 'Audit Records' section selected. The table displays the following audit records:

ID	Created On	Operation	Action	Correlation Id	Created By	Host	Data
6	2018-09-13T13:51:33.085Z	STREAM	CREATE	sample	N/A	pc36.home	time   log
5	2018-09-13T13:20:49.878Z	STREAM	DELETE	mytest	N/A	pc36.home	time   log
4	2018-09-13T13:20:49.872Z	STREAM	UNDEPLOY	mytest	N/A	pc36.home	time   log
3	2018-09-13T13:20:39.975Z	STREAM	CREATE	mytest	N/A	pc36.home	time   log
2	2018-09-13T12:53:30.793Z	STREAM	DEPLOY	foobar	N/A	pc36.home	{"streamDefinitionDslText": "time   log", "deploymentProperties": {}}
1	2018-09-13T12:30:28.932Z	STREAM	CREATE	foobar	N/A	pc36.home	time   log

Items per page: 30 | 1-6 audit records of 6 audit records

## 并发任务启动限制

Spring Cloud Data Flow 允许您强制执行最大数量的并发运行任务，以防止计算资源饱和。可以通过设置 `spring.cloud.dataflow.task.maximum-concurrent-tasks` 属性来配置此限制。默认值为 20。您还可以通过 REST 端点 `/tasks/executions/current` 检索当前正在执行的任务数。新的 `tasklauncher-dataflow` 应用程序利用此功能仅在并发任务数低于最大值时才启动任务。该功能也是正在开发的新 FTP 摄取示例应用程序的核心。

## 流和任务校验

新的 shell 命令 `stream validate` 和 `task validate` 将验证流或任务应用程序资源是否有效且可访问。这可以避免在部署时获得异常。使用 UI 进行验证即将发布。

## 强制升级流应用

强制升级！升级流应用时，您现在可以使用选项 `--force` 来部署当前部署的应用程序的新实例，即使没有更改应用程序或部署属性也是如此。在启动时由应用程序本身获取配置信息（例如从 Spring Cloud Config Server 获取）时，需要此行为。您可以使用选项 `--app-names` 指定要强制升级的应用程序。如果未指定任何应用程序名称，则将强制升级所有应用程序。您还可以将 `--force` 和 `--app-names` 选项与 `--properties` 或 `--propertiesFile` 选项一起指定。

## 接下来计划

1.7 版本将是 1.x 系列中的最后一个次要版本。我们将继续开发 2.0，它将删除“经典”模式并升级到基于 Boot 2.1 的技术栈。其他功能也计划好了，敬请关注。与此同时，期待一些博客文章讨论使用 SCDF 进行 FTP 文件摄取，与 Python 的互操作性以及基于 Spring Cloud Function 的应用程序组合。

## 保持联系

与往常一样，我们欢迎反馈和贡献，请通过 [Stackoverflow](#) 或 [GitHub](#) 或通过 [Gitter](#) 与我们联系。

相关文章：

- [Spring Cloud Data Flow 1.7 RC1 发布](#)
- [Spring Cloud Data Flow 1.7 M1 发布](#)
- [Spring Cloud Data Flow 1.6 正式版发布](#)

NOTE

原文：<https://spring.io/blog/2018/10/25/spring-cloud-data-flow-1-7-ga-released>

翻译：春之雨

说明：版权归原作者所有，翻译仅供学习参考。欢迎反馈和讨论，感谢阅读。

# Spring Cloud Data Flow 1.6 正式版发布

Spring Cloud Data Flow 团队宣布 Spring Cloud Data Flow 1.6 正式版发布。

与此同时，开发团队正在制定 1.7 计划，在 GitHub 上可以了解更多详情。其中一个重点领域是支持 Spring Cloud Function 编程模型，这也有助于简化应用程序启动器和自定义业务逻辑的组合。

1.6 版本有以下几个亮点：

- 任务调度在 PCF 上的原生集成支持
- 仪表板改进
- Kubernetes 支持增强
- 组合任务运行器安全性
- DSL 和部署属性解析改进
- 批处理数据库结构的优化

# Spring Cloud Stream Fishtown.RC2 /2.1.0.RC2 发布

Oleg Zhurakousky 在博客宣布了 Spring Cloud Stream Fishtown.RC2 /2.1.0.RC2 发布。

Spring Cloud Stream Fishtown 2.1.0.RC2 可以在 Spring Milestone 仓库获取到。

作为 2.1.0.RC1 的后续版本，此版本主要包含一些次要增强功能和错误修复。  
目前来看，这应该是几周后 2.1.0.RELEASE 正式版本发布之前的最后一个候选版本。

NOTE

如果应用程序是从 Spring Initializr 创建的，那么它们需要在 spring-cloud BOM 声明之前在 maven 依赖关系管理中添加此 BOM 片段，否则您将使用的是最新的快照版本：

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-stream-dependencies</artifactId>
    <version>Fishtown.RC2</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

[项目主页](#) | [Github](#) | [Gitter](#) | [Stackoverflow](#) | [例子](#)

NOTE

原文：<https://spring.io/blog/2018/11/19/spring-cloud-stream-fishtown-rc2-2-1-0-rc2-release-announcement>  
作者：Oleg Zhurakousky  
编译：春之雨

# Spring Batch 发布说明

轻量级、全面的批处理框架，旨在开发对企业系统日常运营至关重要的强大批处理应用程序。Spring Batch 提供了可重复使用的功能，这些功能对于处理大量记录至关重要，包括日志记录、跟踪、事务管理、作业处理统计、作业重启、跳过和资源管理。它还提供更高级的技术服务和功能，通过优化和分区技术实现极高容量和高性能的批处理作业。简单和复杂的大批量批处理作业可以高度可扩展的方式利用框架来处理大量信息。

其特性如下：

- 事务管理
- 基于块的处理
- 声明性 I/O
- 启动/停止/重新启动
- 重试/跳过
- 基于 Web 的管理界面

表格 7. Spring Batch 2018 年版本发布记录

版本	发布时间	发布说明
4.1.0.RELEASE	2018-10-29 20:10:07	Spring Batch 4.1.0 发布
4.1.0.RC1	2018-09-21 23:16:16	Spring Batch 4.1.0.RC1 发布
4.1.0.M3	2018-08-31 22:47:59	Spring Batch 4.1.0.M3 发布
4.1.0.M2	2018-07-13 19:21:54	
4.1.0.M1	2018-05-31 14:38:24	
3.0.9.RELEASE	2018-03-07 14:52:55	
4.0.1.RELEASE	2018-03-06 21:50:10	

# Spring Batch 4.1.0 发布

Mahmoud Ben Hassine在博客宣布了 Spring Batch 4.1 发布。

## Spring Batch 4.1 有什么新特性

这个版本新增了以下特性：

- 一个新的 `@SpringBatchTest` 注解，用于简化测试批处理组件
- 一个新的 `@EnableBatchIntegration` 注解，用于简化远程分块和分区配置
- 支持以JSON格式读写数据
- 支持使用Bean Validation API验证项目
- 支持JSR-305注解
- `FlatFileItemWriterBuilder` API的增强功能

### `@SpringBatchTest`注解

Spring Batch提供了一些不错的实用程序类（例如 `JobLauncherTestUtils` 和 `JobRepositoryTestUtils`）以及测试执行监听器（`StepScopeTestExecutionListener` 和 `JobScopeTestExecutionListener`）来测试批处理组件。但是，按顺序要使用这些实用程序，必须显式配置它们。此版本介绍一个名为 `@SpringBatchTest` 的新注解，它自动添加实用程序bean和监听测试上下文并使其可用于自动装配，如以下示例所示：

```

@RunWith(SpringRunner.class)
@SpringBatchTest
@ContextConfiguration(classes = {JobConfiguration.class})
public class JobTest {

    @Autowired
    private JobLauncherTestUtils jobLauncherTestUtils;

    @Autowired
    private JobRepositoryTestUtils jobRepositoryTestUtils;

    @Before
    public void clearMetadata() {
        jobRepositoryTestUtils.removeJobExecutions();
    }

    @Test
    public void testJob() throws Exception {
        // given
        JobParameters jobParameters =
            jobLauncherTestUtils.getUniqueJobParameters();

        // when
        JobExecution jobExecution =
            jobLauncherTestUtils.launchJob(jobParameters);

        // then
        Assert.assertEquals(ExitStatus.COMPLETED,
                           jobExecution.getExitStatus());
    }

}

```

### @EnableBatchIntegration 注解

设置远程分块作业需要定义多个bean：

- 用于从消息中间件（JMS、AMQP和其他）获取连接的连接工厂
- **MessagingTemplate**，用于将请求从主服务器发送给工作程序，然后再返回

- Spring Integration的输入通道和输出通道，用于从消息中间件获取消息
- 在主服务器（master）上的特殊内容编写器（`ChunkMessageChannelItemWriter`），它知道如何向工作端（workers）发送数据块以进行处理和写入
- 工作端的消息监听器（`ChunkProcessorChunkHandler`）用于从主服务器接收数据

乍一看，这可能有点令人生畏。此版本引入了一个名为 `@EnableBatchIntegration` 的新注解以及新的API（`RemoteChunkingMasterStepBuilder` 和 `RemoteChunkingWorkerBuilder`）来简化配置。以下示例展示如何使用新注解和API：

```

@Configuration
@EnableBatchProcessing
@EnableBatchIntegration
public class RemoteChunkingAppConfig {

    @Autowired
    private RemoteChunkingMasterStepBuilderFactory masterStepBuilderFactory;

    @Autowired
    private RemoteChunkingWorkerBuilder workerBuilder;

    @Bean
    public TaskletStep masterStep() {
        return this.masterStepBuilderFactory
            .get("masterStep")
            .chunk(100)
            .reader(itemReader())
            .outputChannel(outgoingRequestsToWorkers())
            .inputChannel(incomingRepliesFromWorkers())
            .build();
    }

    @Bean
    public IntegrationFlow worker() {
        return this.workerBuilder
            .itemProcessor(itemProcessor())
            .itemWriter(itemWriter())
            .inputChannel(incomingRequestsFromMaster())
            .outputChannel(outgoingRepliesToMaster())
            .build();
    }

    // Middleware beans setup omitted
}

```

这个新的注解和新的构造器负责配置基础架构bean的繁重工作。您现在可以在工作方（worker）轻松配置主（master）步骤和Spring Integration 流程。

就像远程分块简化配置一样，此版本还引入了新的API来简化远程分区设置：[RemotePartitioningMasterStepBuilder](#) 和 [RemotePartitioningWorkerStepBuilder](#)。如果存在

@EnableBatchIntegration，则可以在配置类中自动装配它们，如以下示例所示：

```
@Configuration
@EnableBatchProcessing
@EnableBatchIntegration
public class RemotePartitioningAppConfig {

    @Autowired
    private RemotePartitioningMasterStepBuilderFactory masterStepBuilderFactory;

    @Autowired
    private RemotePartitioningWorkerStepBuilderFactory workerStepBuilderFactory;

    @Bean
    public Step masterStep() {
        return this.masterStepBuilderFactory
            .get("masterStep")
            .partitioner("workerStep", partitioner())
            .gridSize(10)
            .outputChannel(outgoingRequestsToWorkers())
            .inputChannel(incomingRepliesFromWorkers())
            .build();
    }

    @Bean
    public Step workerStep() {
        return this.workerStepBuilderFactory
            .get("workerStep")
            .inputChannel(incomingRequestsFromMaster())
            .outputChannel(outgoingRepliesToMaster())
            .chunk(100)
            .reader(itemReader())
            .processor(itemProcessor())
            .writer(itemWriter())
            .build();
    }

    // Middleware beans setup omitted
}
```

## 对JSON的支持

此版本引入了一个新的数据读取器（JsonItemReader），它可以按以下格式读取JSON数据：

```
[  
 {  
   "isin": "123",  
   "quantity": 1,  
   "price": 1.2,  
   "customer": "foo"  
 },  
 {  
   "isin": "456",  
   "quantity": 2,  
   "price": 1.4,  
   "customer": "bar"  
 }  
 ]
```

与用于 XML 的 [StaxEventItemReader](#) 类似，新的 [JsonItemReader](#) 使用流 API 来读取块中的 JSON 对象。Spring Batch 支持两个库：

- [Jackson](#)
- [Gson](#)

要添加其他实现，可以实现 [JsonObjectReader](#) 接口。写 JSON 数据也可以通过 [JsonFileItemWriter](#) 来实现。

## 支持 Bean Validation API

此版本带来了一个名为 [BeanValidatingItemProcessor](#) 的新 [ValidatingItemProcessor](#) 实现，它允许您使用 Bean Validation API (JSR-303) 注解进行验证。例如，请考虑以下 Person 类型：

```

class Person {

    @NotEmpty
    private String name;

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}

```

您可以通过在应用程序上下文中声明 `BeanValidatingItemProcessor` bean 来验证项目，并在面向块（chunk-oriented）的步骤中将其注册为处理器，如以下示例所示：

```

@Bean
public BeanValidatingItemProcessor<Person> beanValidatingItemProcessor()
    throws Exception {
    BeanValidatingItemProcessor<Person> beanValidatingItemProcessor
        = new BeanValidatingItemProcessor<>();
    beanValidatingItemProcessor.setFilter(true);

    return beanValidatingItemProcessor;
}

```

## 支持 JSR-305

这个版本添加了对 JSR-305 注解的支持。利用 Spring Framework 的 `Null-safety` 注解，并在 Spring Batch 的所有公共 API 中添加它们。

这些注解不仅在使用 Spring Batch API 时强制执行 null 安全性，而且还可以由 IDE 用于提供与可空性相关的有用信息。例如，如果用户想要实现 `ItemReader` 接口，那么任何支持 JSR-305 注解的 IDE 都将生成如下内容：

```
public class MyItemReader implements ItemReader<String> {  
  
    @Nullable  
    public String read() throws Exception {  
        return null;  
    }  
  
}
```

在 `read` 方法添加 `@Nullable` 注解，明确表示此方法可能返回 `null`。这正如在 Javadoc 中说明的那样，即当数据源处理完成时，`read` 方法应返回 `null`。

#### FlatFileItemWriterBuilder 功能增强

此版本中添加的另一个小功能是简化了文件写入的配置。具体来说，这些更新简化了分隔和固定宽度文件的配置。以下是更改前后的示例。

```

// Before
@Bean
public FlatFileItemWriter<Item> itemWriter(Resource resource) {
    BeanWrapperFieldExtractor<Item> fieldExtractor =
        new BeanWrapperFieldExtractor<Item>();
    fieldExtractor.setNames(new String[] {"field1", "field2", "field3"});
    fieldExtractor.afterPropertiesSet();

    DelimitedLineAggregator aggregator = new DelimitedLineAggregator();
    aggregator.setFieldExtractor(fieldExtractor);
    aggregator.setDelimiter(":");

    return new FlatFileItemWriterBuilder<Item>()
        .name("itemWriter")
        .resource(resource)
        .lineAggregator(aggregator)
        .build();
}

// After
@Bean
public FlatFileItemWriter<Item> itemWriter(Resource resource) {
    return new FlatFileItemWriterBuilder<Item>()
        .name("itemWriter")
        .resource(resource)
        .delimited()
        .delimiter(":")
        .names(new String[] {"field1", "field2", "field3"})
        .build();
}

```

## 其他一些优化

除了所有这些新功能外，此版本还修复了一些错误，并通过 Github 上的 Pull Requests 收集了社区提供的大量增强功能。我们要感谢所有贡献者报告问题、建议功能和贡献代码以使此版本得以发布！可以在 [此处](#) 找到此版本的完整变更记录。

## 反馈

你可以使用即将推出的 Spring Boot 2.1 GA 来使用 Spring Batch 4.1 GA。我们期待您对 [StackOverflow](#)、[Gitter](#) 或 [JIRA](#) 的反馈。



# Spring Batch 4.1.0.RC1 发布

Spring Batch 4.1.0.RC1 发布，本次发布带来了如下内容。

在这个版本中，我们主要致力于 Spring Batch 能够在 Java 8、9、10 和 11 上进行构建和正确运行！此版本基于 [Spring Framework 5.1 GA](#) 以及最新版本的 Spring Integration，Spring AMQP 和 Spring Data。有关更改的完整列表，请参阅 [更改日志](#)。

我们计划是在 10 月底之前发布 Spring Batch 4.1 GA 以及 Spring Boot 2.1 GA。重点是让这个候选版本尽可能稳定，所以请通过测试新功能并提交有关 [JIRA](#)，[StackOverflow](#) 或 [Gitter](#) 的反馈来帮助我们。您可以使用 Spring Boot 2.1.0.M4 使用 Spring Batch 4.1.0.RC1。

[项目首页](#) | [项目源码](#) | [参考手册](#)

相关文章：[Spring Batch 4.1.0.M3 发布](#)

# Spring Batch 4.1.0.M3 发布

Spring Batch 4.1.0.M3 发布，本次发布带来了如下特性：

## 支持 JSR-305

这个里程碑的主题是添加对 JSR-305 注解的支持。利用 Spring Framework 的 **Null-safety** 注解，并在 Spring Batch 的所有公共 API 中添加它们。

这些注解不仅在使用 Spring Batch API 时强制执行 null 安全性，而且还可以由 IDE 用于提供与可空性相关的有用信息。例如，如果用户想要实现 **ItemReader** 接口，那么任何支持 JSR-305 注解的 IDE 都将生成如下内容：

```
public class MyItemReader implements ItemReader<String> {  
  
    @Nullable  
    public String read() throws Exception {  
        return null;  
    }  
  
}
```

在 **read** 方法添加 **@Nullable** 注解，明确表示此方法可能返回 **null**。这正如在 Javadoc 中说明的那样，即当数据源处理完成时，**read** 方法应返回 **null**。

## FlatFileItemWriterBuilder 功能增强

此版本中添加的另一个小功能是简化了文件写入的配置。具体来说，这些更新简化了分隔和固定宽度文件的配置。以下是更改前后的示例。

```
// Before
@Bean
public FlatFileItemWriter<Item> itemWriter(Resource resource) {
    BeanWrapperFieldExtractor<Item> fieldExtractor =
        new BeanWrapperFieldExtractor<Item>();
    fieldExtractor.setNames(new String[] {"field1", "field2", "field3"});
    fieldExtractor.afterPropertiesSet();

    DelimitedLineAggregator aggregator = new DelimitedLineAggregator();
    aggregator.setFieldExtractor(fieldExtractor);
    aggregator.setDelimiter(";");
    return new FlatFileItemWriterBuilder<Item>()
        .name("itemWriter")
        .resource(resource)
        .lineAggregator(aggregator)
        .build();
}

// After
@Bean
public FlatFileItemWriter<Item> itemWriter(Resource resource) {
    return new FlatFileItemWriterBuilder<Item>()
        .name("itemWriter")
        .resource(resource)
        .delimited()
        .delimiter(";")
        .names(new String[] {"field1", "field2", "field3"})
        .build();
}
```

## 其他一些优化

- 通过继承 **DefaultBatchConfigurer** 提供自定义事务管理器的功能
- 修复某些方法名称中的不一致问题

## 反馈

有关更改的完整列表，请查看 [更改记录](#)。这是第一个 RC 之前的最后一个里程碑版本！

[项目主页](#) | [项目源码](#) | [参考手册](#)

# Spring Integration 发布说明

Spring Integration 扩展 Spring 编程模型以支持众所周知的企业集成模式。Spring Integration 在基于 Spring 的应用程序中实现轻量级消息传递，并支持通过声明适配器与外部系统集成。与 Spring 对远程处理，消息传递和调度的支持相比，这些适配器提供了更高级别的抽象。

Spring Integration 的主要目标是提供一个简单的模型来构建企业集成解决方案，同时保持关注点的分离，这对于生成可维护、可测试的代码至关重要。

Spring Cloud Stream 项目建立在 Spring Integration 之上，其中 Spring Integration 用作消息驱动的微服务的引擎。

表格 8. Spring Integration 2018 年版本发布记录

版本	发布时间	发布说明
5.1.1.RELEASE	2018-11-28 20:54:21	
5.0.10.RELEASE	2018-11-28 20:53:40	
4.3.18.RELEASE	2018-11-21 21:59:58	
5.1.0.RELEASE	2018-10-29 17:04:30	Spring Integration 5.1.0 发布
5.1.0.RC2	2018-10-15 20:24:43	
5.0.9.RELEASE	2018-10-15 18:15:22	
5.1.0.RC1	2018-09-21 17:31:15	
5.0.8.RELEASE	2018-09-10 17:48:48	
5.1.0.M2	2018-08-21 19:07:13	
5.1.0.M1	2018-07-30 13:28:23	
5.0.7	2018-07-30 13:25:44	
5.0.6	2018-06-14 15:31:06	
4.3.17	2018-06-14 15:29:02	
5.0.5	2018-06-14 15:27:46	
4.3.16	2018-05-08 20:59:10	
5.0.4.RELEASE	2018-04-04 20:20:09	
4.3.15.RELEASE	2018-04-04 20:18:32	
5.0.3.RELEASE	2018-02-28 21:51:02	
5.0.2.RELEASE	2018-02-28 21:46:59	
4.2.13.RELEASE	2018-02-28 21:49:53	
5.0.1.RELEASE	2018-01-29 22:29:48	
4.3.14.RELEASE	2018-01-29 22:44:48	

# Spring Integration 5.1.0 发布

Artem Bilan在博客上宣布了， Spring Integration 5.1.0 GA 已发布！

现在已经可以从Maven Central, JCenter 和 Spring release仓库下载到。

```
compile "org.springframework.integration:spring-integration-core:5.1.0.RELEASE"
```

首先，我要感谢所有社区成员对框架的持续积极贡献！

除了常规依赖项升级，错误修复和内部性能改进之外，这个版本中还引入了一些值得注意的新功能：

## BoundRabbitChannelAdvice

对于严格排序的消息，可在发布时使用 `BoundRabbitChannelAdvice` 作为 `MessageHandler` 的建议，以允许在同一个线程绑定的 `Channel` 中执行所有下游AMQP操作。通常与分离器（splitter）或其他能够发送多条消息的机制一起使用：

```
@Bean
public IntegrationFlow flow(RabbitTemplate template) {
    return IntegrationFlows.from(Gateway.class)
        .split(s -> s.delimiters(","))
        .advice(new BoundRabbitChannelAdvice(template))
        .<String, String>transform(String::toUpperCase)
        .handle(Amqp.outboundAdapter(template).routingKey("rk"))
        .get();
}
```

## 响应式轮询（Reactive Polling）

如果将 `SourcePollingChannelAdapter` 或 `PollingConsumer` 配置为 `FluxMessageChannel` 类型的输出通道 `outputChannel`，则调度程序不会执行轮询任务，而是基于时间 `trigger.nextExecutionTime()` 作为后续 `Mono.delay()` 被 `Flux.generate()` 调用。因此，真正的轮询源按需执行，以满足 `FluxMessageChannel` 传播的下游背压（downstream back-pressure）。对于最终配置，`PollerMetadata` 选项保持不变且透明。

## Java DSL `fluxTransform()`

`IntegrationFlowDefinition` 现在提供了新的 `fluxTransform()` 运算符，该运算符接受一个 `Function`，用于响应式的转换输入消息的 `Flux`。`fluxTransform()` 的实现完全基于Reactor的 `Flux.transform()` 运算符，如果函数不处理已经存在的消息，则将请求标头复制到回复消息。调用内部包含输入和输出 `FluxMessageChannel` 实例：

```
IntegrationFlow integrationFlow = f -> f
    .split()
    .<String, String>fluxTransform(flux -> flux
        .map(Message::getPayload)
        .map(String::toUpperCase))
    .aggregate(a -> a
        .outputProcessor(group -> group
            .getMessages()
            .stream()
            .map(Message::getPayload)
            .map(String.class::cast)
            .collect(Collectors.joining(","))))
    .channel(resultChannel);
```

此外，Java DSL为方便起见提供了更多运算符：`nullChannel()`，`convert(Class <? >)` 和 `logAndReply()`。有关更多信息，请参阅他们的 Javadocs。

## Java Function 改进

`java.util.function` 接口现在充当消息传递端点的一等公民。`Function<?, ?>` 和 `Consumer<?>` 可以在 `@ServiceActivator` 或 `@Transformer` 定义中直接使用，例如：

```
@Bean
@Transformer(inputChannel = "functionServiceChannel")
public Function<String, String> functionAsService() {
    return String::toUpperCase;
}
```

`Supplier<?>` 接口可以简单的与 `@InboundChannelAdapter` 注解一起使用，或者作为 `<int:inbound-channel-adapter>` 的引用 ref：

```
@Bean
@InboundChannelAdapter(value = "inputChannel", poller = @Poller(fixedDelay = "1000"))
public Supplier<String> pojoSupplier() {
    return () -> "foo";
}
```

Kotlin lambdas现在也可以直接用于消息传递端点定义：

```

@Bean
@Transformer(inputChannel = "functionServiceChannel")
fun kotlinFunction(): (String) -> String {
    return { it.toUpperCase() }
}

@Bean
@ServiceActivator(inputChannel = "messageConsumerServiceChannel")
fun kotlinConsumer(): (Message<Any>) -> Unit {
    return { print(it) }
}

@Bean
@InboundChannelAdapter(value = "counterChannel",
    poller = [Poller(fixedRate = "10", maxMessagesPerPoll = "1")])
fun kotlinSupplier(): () -> String {
    return { "baz" }
}

```

## Micrometer

随着升级到Micrometer 1.1版本，框架现在可以自动从 [MeterRegistry](#) 中删除已注册的 meters。当我们开发动态 [IntegrationFlow](#) 并在运行时注册和删除它们时，这变得非常方便。

## JMX MBeans可删除

现在MBean可在运行时（例如通过动态 [IntegrationFlow](#)）被注册，而当它们的bean在运行时被销毁时，也会自动从JMX服务器注册中被删除。

## HTTP动态映射（Dynamic Mapping）

HTTP（和WebFlux）入站端点如果是在运行时声明（例如通过动态 [IntegrationFlow](#)），那么在 [HandlerMapping](#) 中将会自动注册它们的请求映射，同时在它们的 bean 被销毁时也会自动被删除。

## 支持Spring Social Twitter

由于 Spring Social项目即将 [End of Life](#)，我们已经将 [spring-integration-twitter](#) 模块移动到 Spring [Integration Extensions](#)下的独立项目，并刚刚发布了一个 [org.springframework.integration:spring-integration-social-twitter:1.0.0.RELEASE](#)，它也是完全基于Spring Integration 5.1。

## 小结

有关更多变更的内容，请参阅参考手册的“[新增内容](#)”一章。另请参阅[迁移指南](#)以了解此版本中的更改以及如何处理它们。

此版本是Spring Boot 2.1 GA 的基础。

接下来，我们准备将master切换到 5.2 版本，开始开发新功能和有价值的改进！

欢迎通过合适的沟通渠道提供任何反馈，特性创意、评论、错误报告和问题：

[项目主页](#) | [JIRA](#) | [参与贡献](#) | [寻求帮助](#) | [在线群聊](#)

NOTE

原文：<https://spring.io/blog/2018/10/29/spring-integration-5-1-goes-ga>  
作者：Artem Bilan

## Spring Integration、AMQP和Kafka RC1 发布

Spring Framework 5.0.9 修复了 36 个问题单，Spring Framework 4.3.19 修复了 23 个问题单，而 Spring Framework 5.1 RC3 修复了 30 个问题单。

Spring Framework 5.1 RC3 升级了依赖：Reactor Californium RC1、RxJava 2.2 和 JUnit Jupiter 5.3。

接下来，Spring Boot 2.1 M3，2.0.5 和 1.5.16 将会在下周发布，敬请期待！

# Spring Integration for AWS 2.0 发布

## 介绍

**亚马逊网络服务** (Amazon Web Services: AWS) 于2006年推出，通过其云计算平台为企业提供关键的基础设施服务。使用云计算业务可以采用新的业务模式，从而无需计划和投资采购自己的IT基础架构。他们可以使用云服务提供商提供的基础架构和服务，并在使用服务时付费。访问 <http://aws.amazon.com/products/>，了解亚马逊作为其云计算服务的一部分提供的各种产品的更多详细信息。

Spring Integration AWS 扩展，通过 [AWS Java SDK](#) 为 Spring Integration 提供了集成各种AWS服务的适配器。请注意，Spring Integration AWS Extension 基于 Spring Cloud AWS 项目。

Spring Integration AWS 扩展已支持集成的服务有：

- Amazon Simple Email Service (SES)
- Amazon Simple Storage Service (S3)
- Amazon Simple Queue Service (SQS)
- Amazon Simple Notification Service (SNS)
- Amazon DynamoDB
- Amazon SimpleDB

## 如何使用

在Maven中添加依赖项

如果使用Maven，可以在项目POM的依赖中添加如下，这些组件已经发布到Spring Release和Maven Central了。

```
<dependency>
    <groupId>org.springframework.integration</groupId>
    <artifactId>spring-integration-aws</artifactId>
    <version>2.0.0.RELEASE</version>
</dependency>
```

## 集成 Amazon S3 服务

入站通道适配器

S3 通道适配器 (Channel Adapters) 基于 [AmazonS3](#) 模板和 [TransferManager](#)。

S3 入站通道适配器 (Inbound Channel Adapter)，由 [S3InboundFileSynchronizingMessageSource](#) ([`<int-aws:s3-inbound-channel-adapter>`](#)) 来表示，允许将 S3 对象作为文件从 S3 存储桶拉到本地目录以进行同步。这个适配器与 Spring Integration 的模块 FTP 和 SFTP 中的入站通道适配器完全类似。

Java 配置方式如下：

```

@SpringBootApplication
public static class MyConfiguration {

    @Autowired
    private AmazonS3 amazonS3;

    @Bean
    public S3InboundFileSynchronizer s3InboundFileSynchronizer() {
        S3InboundFileSynchronizer synchronizer = new S3InboundFileSynchronizer(amazonS3());
        synchronizer.setDeleteRemoteFiles(true);
        synchronizer.setPreserveTimestamp(true);
        synchronizer.setRemoteDirectory(S3_BUCKET);
        synchronizer.setFilter(new S3RegexPatternFileListFilter(".*\\.test$"));
        Expression expression = PARSER.parseExpression("#this.toUpperCase() + '.a'");
        synchronizer.setLocalFilenameGeneratorExpression(expression);
        return synchronizer;
    }

    @Bean
    @InboundChannelAdapter(value = "s3FilesChannel", poller = @Poller(fixedDelay = "100"))
    public S3InboundFileSynchronizingMessageSource
    s3InboundFileSynchronizingMessageSource() {
        S3InboundFileSynchronizingMessageSource messageSource =
            new S3InboundFileSynchronizingMessageSource(s3InboundFileSynchronizer());
        messageSource.setAutoCreateLocalDirectory(true);
        messageSource.setLocalDirectory(LOCAL_FOLDER);
        messageSource.setLocalFilter(new AcceptOnceFileListFilter<File>());
        return messageSource;
    }

    @Bean
    public PollableChannel s3FilesChannel() {
        return new QueueChannel();
    }
}

```

使用此配置，你将从 **s3FilesChannel** 接收带有 **java.io.File payload** 的消息，可以定期将 Amazon S3 存储桶中的内容同步到本地目录。

如果使用 XML 的配置，类似这样：

```

<bean id="s3SessionFactory"
      class="org.springframework.integration.aws.support.S3SessionFactory"/>

<int-aws:s3-inbound-channel-adapter channel="s3Channel"
    session-factory="s3SessionFactory"
    auto-create-local-directory="true"
    delete-remote-files="true"
    preserve-timestamp="true"
    filename-pattern="*.txt"
    local-directory="."
    local-filename-generator-expression="#this.toUpperCase() + '.a' + @fooString"
    temporary-file-suffix=".foo"
    local-filter="acceptAllFilter"
    remote-directory-expression="my_bucket">
    <int:poller fixed-rate="1000"/>
</int-aws:s3-inbound-channel-adapter>
```

#### 出站通道适配器

S3 出站通道适配器，由 `S3MessageHandler` (`<int-aws:s3-outbound-channel-adapter>` 和 `<int-aws:s3-outbound-gateway>`) 表示，允许上传、下载、拷贝等 S3 存储桶提供的操作（参见：`S3MessageHandler.Command` 枚举）。

使用 Java 配置：

```

@SpringBootApplication
public static class MyConfiguration {

    @Autowired
    private AmazonS3 amazonS3;

    @Bean
    @ServiceActivator(inputChannel = "s3UploadChannel")
    public MessageHandler s3MessageHandler() {
        return new S3MessageHandler(amazonS3, "myBuck");
    }

}
```

使用此配置，你可以执行 `transferManager.upload()` 操作，使用 `java.io.File` 作为 `payload` 发送消息，其中文件名用作 S3 对象键。

使用XML配置：

```
<bean id="transferManager" class="com.amazonaws.services.s3.transfer.TransferManager"/>

<int-aws:s3-outbound-channel-adapter transfer-manager="transferManager"
    channel="s3SendChannel"
    bucket="foo"
    command="DOWNLOAD"
    key="myDirectory"/>
```

## 出站网关

S3 出站网关（Outbound Gateway），Java 配置使用 `S3MessageHandler` 带构造函数参数 `produceReply = true` 定义，XML 配置通过 `<int-aws:s3-outbound-gateway>` 来声明。

此网关的“请求-回复”是异步的，并且 `TransferManager` 操作的传输结果将发送到 `outputChannel`，假设传输进度跟踪在下游传输流中。

你可以用 `S3ProgressListener` 来跟踪传输进度，也可以在下游传输流中将其填充到返回的 `Transfer` 中。

如需了解更多信息，请查看 `S3MessageHandler` 的 Java 文档，以及 `<int-aws:s3-outbound-channel-adapter>` 和 `<int-aws:s3-outbound-gateway>` 的描述。

## 这样做的好处

Spring Integration 提供了 Spring 编程模型的扩展，以支持众所周知的企业集成模式（EIP）。

Spring Integration for AWS 2.0 的主要目的是遵循设计良好的 Spring Integration 抽象，在其基础上扩展支持 AWS 服务的能力，提供编程一致的系统集成模式。当你的应用程序部署到 Amazon 云环境中或依赖 AWS 服务时，这些新组件也可用于任何其他分布式任务。例如，AWS S3 入站通道适配器可以将 `DynamoDbMetadataStore` 用于 `S3PersistentAcceptOnceFileListFilter`，以防止在多个分布式应用程序实例中访问相同的已处理资源。`DynamoDbLockRegistry` 可用于应用程序集群中的 Leader Election（领导者选举）。

## 参考资料

Spring Integration: <https://github.com/spring-projects/spring-integration>

Spring Integration Extension for AWS: <https://github.com/spring-projects/spring-integration-aws>

Spring Cloud for Amazon Web Services: <https://cloud.spring.io/spring-cloud-aws/>

# Spring 开发指南

本章是本年度精选的一些 Spring 开发指南。

- Spring 开发指南，带你进入 Spring 的世界
- 安装 Java SDK 8
- 安装 Maven
- 安装 Spring Boot CLI
- Spring Boot CLI 使用详解
- 安装 STS



# Spring开发指南，带你进入Spring的世界

## 概要

本文主要介绍如何学习使用官方的 Spring 开发指南。

[Spring 开发指南](#)是 Spring 官方提供的 Spring 技术的入门级学习指南，内容主题全面覆盖了 Spring 技术栈，每篇指南都遵循一套标准化的学习流程，从新手入门到专题介绍，从基础概念介绍到编码实战，即使你什么都不会，也可以从零开始跟着文档一步步完成编码，最终完成学习这并不是一件难事，在每个指南中同时提供了多种主流构建方式（Maven、Gradle、IDE），项目完整的源代码放在 Github 上可下载，内容与最新版本同步更新，有很好的学习借鉴和指导性，作为初学者的技术入门和了解新技术都是不错的必备之选。

如果有人问我推荐什么资料，我一定首先推荐官方的开发指南，再配合各个框架参考文档，作为入门学习已经足够了。但是让很多人望而却步的是“看着一大篇英文就头疼”，这的确是个问题，这需要一边写着代码还得一边在线翻译资料。作为一个专业的软件开发人，这似乎是必经之路，无论你走过多少路都得回到开始的地方，源代码和参考文档就是那个“最初的地方”。当然，我也能理解，学习一门英语也非“一朝一夕之事”。

## 关于Spring开发指南

Spring开发指南分为三个类型，

- **新手入门**（Getting Started Guides），适合新手入门，一般是使用Spring技术来构建诸如“Hello World”的简单示例，学习时间在15到30分钟
- **专题指南**（Topical Guides），比新手入门的内容广泛，带有独特观点的技术讲解，适合全面的理解技术和框架的架构设计，学习时间在一个小时左右
- **学习教程**（Tutorials），更加深度和主题的企业级应用开发，提供现实世界复杂问题的解决方案，学习时间在2到3小时

在新手入门阶段，往往需要提前了解一些基础知识和概念，这些知识有助于理解所学内容和顺利完成学习任务。比如，JSON、REST、POJO、JavaScript、Tomcat等，除此之外，还需要会安装一些常用Java相关开发工具，比如Java SDK、Git、STS等，这些都会在开发中经常用到。

目前已经提供的新手入门，已有发布有 69 篇，涵盖主题有前端（jQuery、AngularJS、Vaadin）、工具（Maven、Gradle、STS、IntelliJ IDEA.）数据库（MySQL、JPA、Redis、MongoDB、Neo4j）、消息与服务（JMS、Web Service）、云（Google Cloud、Cloud Foundry、网关、配置管理）、测试、安全等等，已经相当全面了。而专题方面则只有一篇是关于Spring Security 的，这个我目前正在写一些有关的文章。教程方面目前也有 5 篇，仍然有更多可完善的内容，尤其是关于 Spring Cloud 方面比较缺乏。

## 学习建议

如果是刚接触到 Spring 技术，建议先从新手入门学起，先跟着示例一步一步将代码完成，这样会有个基本认识，如果有不清楚的地方可以先查找Spring框架对应的参考文档。目前的大部分例子都是直接使用 Spring Boot 进行开发的，这一开始会让你觉得它很简单快速，少量的代码和少量的配置就能立即运行起来，这对于初学者其实是一种诱惑，久而久之会造成一种错觉，认为简单是理所当然的。如果这样去想，后面必定会吃亏踩坑导致心理上受挫。Spring 的项目越来越多，各个框架之间的整合和依赖也越来越密切。自动装配会减少配置，但也让配置变得无处不在（条件和规则只是隐藏在代码之中），出错之后定位和解决的难度增加。各种 starters 只是屏蔽了第三方项目的直接依赖，但是这种依赖依然存在，你仍然需要知道依赖的是什么。所以在学习的过程中，还是要学习其他框架和库，了解它们仍然很有必要。最基础的技术比如 Java EE 的一些规范，Spring Framework 等等，还有部分人认为会 Spring Boot、Spring Cloud 就不需要学习 Spring

Framework，这当然是不对的。在领进门之后，剩下的是需要时间，在不断的项目实战和学习摸索中领悟 “The Spring Way” !

## 文档和代码下载使用

官方的开发指南都托管在 [Github](#)上，所有的代码遵守 [ASLv2](#)，而所有的文档遵守[Attribution, NoDerivatives creative commons license](#)，任何人都可以阅读、下载包括分发，但是禁止演绎，也就是不允许修改和翻译。

## 书籍推荐

关于 Spring 技术的书籍，已经有非常多了，可以参考我写的另外一篇文章：[最全 Spring 书单，送给爱学习的你](#)，后续会有更多书评来帮助大家选择和快速阅读学习，敬请关注。

# 安装 Java SDK 8

## 概要

本文主要讲述如何安装JDK 8。

### 在Oracle官方下载安装程序

打开浏览器，在 Oracle 网站，通过 [Java 8 下载页面](#)来下载使用操作系统对应的安装程序。

- 在Windows系统上安装

运行安装程序jdk-8u172-windows-x64.exe，根据向导设置并下一步，记得将安装路径修改为 C:\Java\jdk1.8.0，因为在Windows系统下，Java的安装路径最好不要包括空格。

**NOTE** JAVA\_HOME 环境变量必须正确配置。  
可以通过右键单击我的电脑，选择属性菜单，然后在高级页签中选择环境变量。添加环境变量 JAVA\_HOME即可。

### 使用SDKMAN 安装

**SDKMAN** 是一个 JVM 上各种语言 SDK 和工具包的安装管理工具，提供了一些常用 SDK 的安装和升级功能，类似 Ruby 的**RVM**或 NodeJS 下的 nvm，可以同时安装多个版本并支持随时切换使用。支持 Java、Scala、Kotlin、Groovy、Grails、Ant、Maven、Gradle 等流行SDK和工具。

```
$ sdk i java 8.0.172-zulu
```

安装完并配置好环境变量后，可以测试下是否成功，打开一个控制台窗口，输入如下命令：

```
$ java version
openjdk version "1.8.0_172"
OpenJDK Runtime Environment (Zulu 8.30.0.1-win64) (build 1.8.0_172-b01)
OpenJDK 64-Bit Server VM (Zulu 8.30.0.1-win64) (build 25.172-b01, mixed mode)
```

# 安装 Maven

## 概要

本文主要介绍Maven的多种安装方式。

Maven 是一个 Java 平台上非常流行的项目构建和管理工具，支持项目的编译、打包、测试、第三方依赖包管理，在很多知名的开源项目和企业中使用。同时也能和 Ant 进行集成，保证一些使用 Ant 的旧项目能够顺利迁移，同时也提供一些比较灵活性。

## 下载离线安装包手动安装

当前使用的版本是 3.5.2，去[Maven官方下载](#) `apache-maven-3.5.2-bin.zip` 或者 `apache-maven-3.5.2-bin.tar.gz`，然后解压到本地目录并设置好 `M2_HOME` 和 `Path` 环境变量即可。

## 使用 SDKMAN 安装

[SDKMAN](#) 是一个 JVM 上各种语言 SDK 和工具包的安装管理工具，它的前身是 GVM。

```
$ sdk install maven
```

## 使用OSX Homebrew安装

[Homebrew](#)是专为 macOS 系统设计的软件包安装管理工具。

在终端控制台上执行下面的命令来安装Maven：

```
$ brew install maven
```

## 使用 Scoop 安装

[Scoop](#) 是使用PowerShell 语言为 Windows 系统开发的一个命令行的安装管理工具，功能与OSX Homebrew类似。Scoop 的详细安装请参考 [官方安装说明](#)。

在命令行窗口执行下面的命令来安装 Maven 如下：

```
$ scoop install maven
```

安装之后，检查下安装是否正确：

```
$ mvn --version
Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T15:58:13+08:00)
```

# 安装 Spring Boot CLI

## 概要

本文主要介绍Spring Boot的多种安装方式。

Spring Boot CLI 是一个快速创建原型项目的命令行工具。

### 下载离线安装包手动安装

下载 Spring Boot CLI 安装包，然后解压到本地目录并设置好 Path 环境变量即可。

### 使用 SDKMAN 安装

SDKMAN 是一个 JVM 上各种语言 SDK 和工具包的安装管理工具，它的前身是 GVM。

```
$ sdk install springboot  
$ spring --version  
Spring Boot v1.5.13.RELEASE
```

### 使用OSX Homebrew安装

Homebrew 是专为 macOS 系统设计的软件包安装管理工具。

在终端控制台上执行下面的命令来安装 Spring Boot：

```
$ brew tap pivotal/tap  
$ brew install springboot
```

### 使用Scoop安装

Scoop 是使用 PowerShell 语言为 Windows 系统开发的一个命令行的安装管理工具，功能与 OSX Homebrew 类似。Scoop 的详细安装请参考 [官方安装说明](#)。

在命令行窗口执行下面的命令来安装Spring Boot如下：

```
> scoop bucket add extras  
> scoop install springboot
```

安装之后，检查下安装是否正确：

```
$ spring --version  
Spring Boot v1.5.13.RELEASE
```



# Spring Boot CLI 使用详解

## 概要

本文详细介绍了 Spring Boot CLI 的命令以及在实际开发中如何使用。

在之前的文章中安装Spring Boot CLI，我们介绍了几种常见的快速安装方式。接下来这个章节，我们会详细介绍它的命令和在实际开发中的使用。

Spring Boot CLI 是一个快速创建原型项目的命令行工具。对大多数开发人员来说，相比IDE的菜单选择和向导界面，会更倾向使用命令行来处理一些基础工作，比如创建项目、运行测试、编译打包等等，Spring Boot CLI 就是为这些人准备的。

## 常用命令及参数说明

打开一个终端控制台窗口，我们键入 `spring` 命令，就会看到控制台打印出了该命令的帮助说明。

命令	功能说明
run	运行一个Groovy脚本程序
test	测试一个Groovy脚本程序
grab	从远程仓库下载Groovy脚本程序需要的依赖包
jar	将一个Groovy脚本程序打包成可执行的JAR文件
war	将一个Groovy脚本程序打包成可执行的WAR文件
install	安装依赖包并下载到lib/ext目录
uninstall	卸载依赖包并从lib/ext目录删除
init	初始创建一个新工程，和start.spring.io提供的页面功能一样

```
$ spring
```

```
usage: spring [--help] [--version]
               <command> [<args>]
```

Available commands are:

```
run [options] <files> [--] [args]
Run a spring groovy script
```

```
test [options] <files> [--] [args]
Run a spring groovy script test
```

grab

Download a spring groovy script's dependencies to ./repository

jar [options] <jar-name> <files>

Create a self-contained executable jar file from a Spring Groovy script

war [options] <war-name> <files>

Create a self-contained executable war file from a Spring Groovy script

install [options] <coordinates>

Install dependencies to the lib/ext directory

uninstall [options] <coordinates>

Uninstall dependencies from the lib/ext directory

init [options] [location]

Initialize a new project using Spring Initializr ([start.spring.io](http://start.spring.io))

shell

Start a nested shell

Common options:

-d, --debug Verbose mode

Print additional status information for the command you are running

See 'spring help <command>' for more information on a specific command.

## 运行 Groovy 脚本程序

接下来我们重点讲解 **run** 和 **init** 这两个命令的用法。

在终端控制台上执行下面的命令来查看下帮助：

```
$ spring help run
spring run - Run a spring groovy script
```

usage: spring run [options] <files> [--] [args]

Option	Description
--autoconfigure [Boolean]	Add autoconfigure compiler transformations (default: true)
--classpath, -cp	Additional classpath entries
--no-guess-dependencies	Do not attempt to guess dependencies
--no-guess-imports	Do not attempt to guess imports
-q, --quiet	Quiet logging
-v, --verbose	Verbose logging of dependency resolution
--watch	Watch the specified file for changes

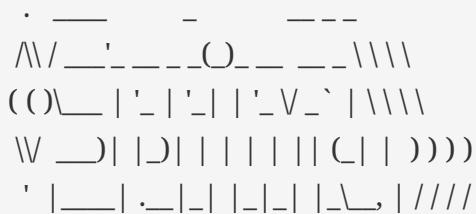
接着，我们简单编写一段Groovy脚本程序：[hello.groovy](#)，代码如下

```
@RestController
class WebApplication {

    @RequestMapping("/")
    String home() {
        "Hello World!"
    }
}
```

然后试着执行命令：[spring run hello.groovy](#)，是的，没错，它的确可以跑起来，这真是太棒了！

Resolving dependencies.....



===== | \_ | ===== | \_/=/\\_\\_

:: Spring Boot :: (v1.5.13.RELEASE)

```
2018-06-03 11:25:23.632 INFO 10692 --- [    runner-0] o.s.boot.SpringApplication      : Starting application on Jill with PID 10692 (started by rain in \Development\springdev\code)
2018-06-03 11:25:23.647 INFO 10692 --- [    runner-0] o.s.boot.SpringApplication      : No active profile set, falling back to default profiles: default
2018-06-03 11:25:24.471 INFO 10692 --- [    runner-0]
ationConfigEmbeddedWebApplicationContext : Refreshing
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@7d093264: startup date [Sun Jun 03 11:25:24 CST 2018]; root of context hierarchy
2018-06-03 11:25:26.742 INFO 10692 --- [    runner-0]
s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2018-06-03 11:25:26.810 INFO 10692 --- [    runner-0] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2018-06-03 11:25:26.812 INFO 10692 --- [    runner-0]
org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.31
2018-06-03 11:25:27.031 INFO 10692 --- [ost-startStop-1]
org.apache.catalina.loader.WebappLoader : Unknown loader
org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentClassLoader@73e0e880 class
org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentClassLoader
2018-06-03 11:25:27.123 INFO 10692 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]      :
Initializing Spring embedded WebApplicationContext
2018-06-03 11:25:27.123 INFO 10692 --- [ost-startStop-1] o.s.web.context.ContextLoader      :
Root WebApplicationContext: initialization completed in 2652 ms
2018-06-03 11:25:27.448 INFO 10692 --- [ost-startStop-1]
o.s.b.w.servlet.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]
2018-06-03 11:25:27.457 INFO 10692 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean :
Mapping filter: 'characterEncodingFilter' to: [//*]
2018-06-03 11:25:27.458 INFO 10692 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean :
Mapping filter: 'hiddenHttpMethodFilter' to: [//*]
2018-06-03 11:25:27.458 INFO 10692 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean :
Mapping filter: 'httpPutFormContentFilter' to: [//*]
2018-06-03 11:25:27.458 INFO 10692 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean :
Mapping filter: 'requestContextFilter' to: [//*]
2018-06-03 11:25:28.188 INFO 10692 --- [    runner-0]
s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice:
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationCo
```

```

ntext@7d093264: startup date [Thu Jul 19 11:25:24 CST 2018]; root of context hierarchy
2018-06-03 11:25:28.336 INFO 10692 --- [    runner-0]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/]}" onto public java.lang.String
WebApplication.home()
2018-06-03 11:25:28.344 INFO 10692 --- [    runner-0]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public
org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>>
org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.Http
tpServletRequest)
2018-06-03 11:25:28.345 INFO 10692 --- [    runner-0]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto
public org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.ht
tp.HttpServletRequest,javax.servlet.http.HttpServletResponse)
2018-06-03 11:25:28.428 INFO 10692 --- [    runner-0]
o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler of
type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-06-03 11:25:28.429 INFO 10692 --- [    runner-0]
o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-06-03 11:25:28.531 INFO 10692 --- [    runner-0]
o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler
of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-06-03 11:25:29.935 INFO 10692 --- [    runner-0] o.s.j.e.a.AnnotationMBeanExporter      :
Registering beans for JMX exposure on startup
2018-06-03 11:25:30.026 INFO 10692 --- [    runner-0]
s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2018-06-03 11:25:30.036 INFO 10692 --- [    runner-0] o.s.boot.SpringApplication      :
Started application in 7.352 seconds (JVM running for 39.869)

```

## 快速创建一个新工程

如果在 IDE 中创建一个 Spring 新工程，大概至少需要四步，首先打开新建向导 [Spring Starter Project](#)，接着输入项目名称、包名，选择语言、类型等，然后选择工程依赖，如 Web、Thymeleaf、DevTools 等，最后检查并确定。这的确太繁琐了，如果还得做好几遍的话。

但是这些使用 Spring Boot CLI 就能轻轻松松完成，打开控制台，输入下面这段命令即可：

```
$ spring init --name hello-world --artifactId hello-world --groupId org.springbooks --language java --boot-version 1.5.13.RELEASE --type maven-project --dependencies web,thymeleaf,devtools --extract
```

这的的确是很简单，对于我们只要记住一些常用的命令参数而已：

对于 `spring init` 的参数，有以下几个：

参数	说明	默认值
name	项目的名称	demo
package-name	项目的包名	com.example.demo
description	项目的描述	Demo project for Spring Boot
artifactId	项目的构件名称	demo
groupId	项目的组织名称	com.example
javaVersion	Java语言的版本	1.8
language	可以选择语言，java、groovy、kotlin	java
packaging	项目打包类型，可选择：jar、war	jar
type	项目类型，可选择：maven-project, maven-build, gradle-project, gradle-build	maven-project
version	项目的版本号	0.0.1-SNAPSHOT
bootVersion	Spring Boot的版本	默认是安装的版本，如：2.0.3.RELEASE

其中大部分参数都有默认值，可以省略暂时不填写。对于 `dependencies`，可以通过以下命令来获取：

```
$ spring init --list
```

对于常用的一些依赖模块，详见下面的表格：

名称	说明
actuator	Production ready features to help you monitor and manage your application
aop	Create your own Aspects using Spring AOP and AspectJ
cache	Spring's Cache abstraction
data-jpa	Java Persistence API including spring-data-jpa, spring-orm and Hibernate

名称	说明
data-mongodb	MongoDB NoSQL Database, including spring-data-mongodb
data-redis	Redis key-value data store, including spring-data-redis
devtools	Spring Boot Development Tools
flyway	Flyway Database Migrations library
freemarker	FreeMarker templating engine
mybatis	Persistence support using MyBatis
mysql	MySQL JDBC driver
security	Secure your application via spring-security
thymeleaf	Thymeleaf templating engine
web	Full-stack web development with Tomcat and Spring MVC

举个例子：通过 CLI 创建新项目

```
$ spring init --name hello-world --artifactId hello-world --groupId org.springbooks --language java --boot-version 1.5.13.RELEASE --type maven-project --dependencies web,thymeleaf,devtools --extract
```

我们继续来看，创建的项目目录如下：

```
├── mvnw
├── mvnw.cmd
└── pom.xml
└── src
    ├── main
    │   ├── java
    │   │   └── org
    │   │       └── springbooks
    │   │           └── helloworld
    │   │               └── HelloWorldApplication.java
    │   └── resources
    │       ├── application.properties
    │       ├── static
    │       └── templates
    └── test
        └── java
            └── org
                └── springbooks
                    └── helloworld
                        └── HelloWorldApplicationTests.java
```

一切就这么容易搞定，接下来就可以写代码了。

# 安装 STS

## 概要

本文主要介绍有关 STS 的工具特性和如何安装。

Spring Tool Suite (简称：STS) 是 Spring Source 开发团队在基于 Eclipse 基础上为 Spring 应用的开发专门定制的一款集成开发环境，对于 Spring 应用的编码实现、调试、部署等各个流程，集成了包括 Pivotal tc Server、Pivotal Cloud Foundry、Git、Maven、AspectJ 等工具，极大提供了开发效率。

主要特性如下：

- 支持 Java 8/9/10
- 集成 Git、Gradle、Maven 等工具
- 更好的识别 Spring 项目，能够识别 Spring 配置文件，诸如命名空间、控制器、Bean 的定义
- Spring 配置文件的校验检查，更早发现错误
- 重构增强，除了对于 Java 代码的重构功能，还支持 Spring 配置文件中的重构，比如 Bean 的重命名
- 代码快速提示，无论是 Java 代码和配置文件都能根据内容快速提示
- 图形化展示项目中 Bean 的依赖关系，支持 Spring Integration、Spring Batch、Spring Webflow
- AOP 最佳支持
- 集成 Cloud Foundry 和 Pivotal tc Server，云部署开发更加简单

## 通过下载安装包来安装

浏览器打开 STS 官方网站 [STS](#)，当前最新版本是：3.9.4，基于 Eclipse 4.7.3a 构建。除了使用完整安装包，如果本地已经安装了 Eclipse，那么还可以通过插件的方式来安装。STS 官方提供了 Eclipse 4.6、4.7、4.8 等几个主流版本的插件可下载本地安装，也可以通过在线安装插件。

The screenshot shows the official website for Spring Tool Suite (STS). At the top, there's a navigation bar with links for 'PROJECTS', 'GUIDES', 'BLOG', and a search icon. Below the header, the 'spring' logo is displayed, followed by the text 'by Pivotal'. A horizontal menu bar labeled 'TOOLS' is visible. The main content area features a section titled 'Spring Tool Suite™' which describes the tool as an Eclipse-based development environment for Spring applications. It includes details about its integration with Pivotal tc Server, Git, Maven, and AspectJ. Below this, another section discusses the developer edition of tc Server and its Spring Insight console. To the right, there's a prominent download button for 'STS (3.9.4.RELEASE for Mac)' and a link to 'See All Versions'. The overall design is clean and professional, using a white background with black and green text.

图 9. STS官方主页

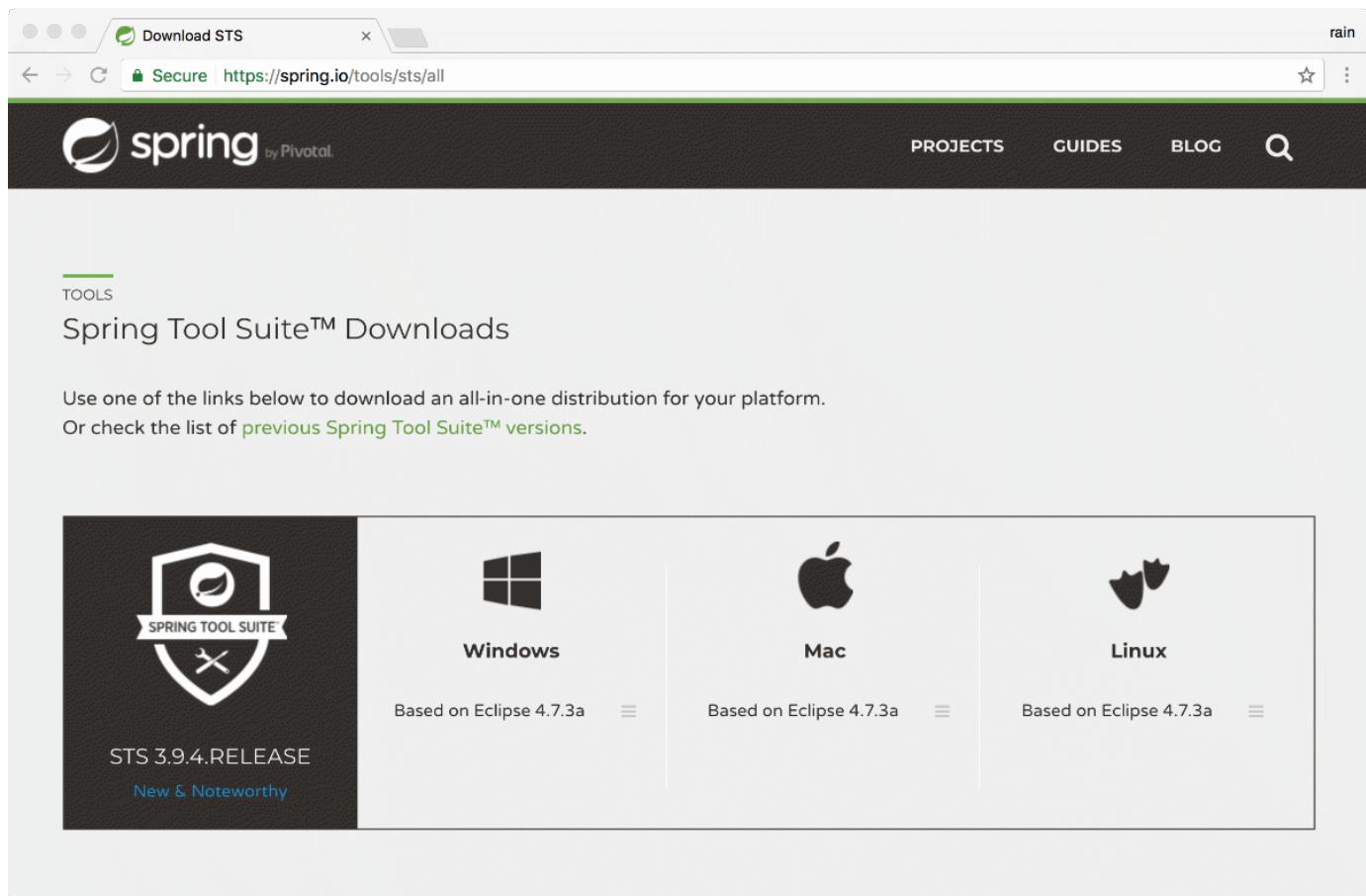


图 10. STS下载页面

## 使用 OSX Homebrew 安装

Homebrew 是专为 macOS 系统设计的软件包安装管理工具。

在终端控制台上执行下面的命令来安装STS：

```
$ brew cask install sts
```

在 Linux 系统上，有 Homebrew 的 Fork 版本：Linuxbrew，可以尝试一下。

## 使用 Scoop 安装 STS

Scoop 是使用 PowerShell 语言为 Windows 系统开发的一个命令行的安装管理工具，功能与 OSX Homebrew 类似。Scoop 的详细安装请参考[官方安装说明](#)。

在命令行窗口执行下面的命令来安装STS如下：

```
> scoop bucket add extras
> scoop install sts
```

# Spring 精选文章

本章是本年度精选的一些 Spring 相关文章。

- 最全 Spring 书单，送给爱学习的你
- Spring Boot 最佳实践
- 10 种保护 Spring Boot 应用程序的绝佳方法

Spring in 2018

# 最全 Spring 书单，送给爱学习的你

送走了炎炎夏日，迎来了金秋九月，正是新生们开学的季节，对于开学最期待的事，莫过于可能会遇到新的老师，结识新的朋友，最重要的是可以拿到新的书本，开始学习新的知识。而对于毕业季开始参加工作的朋友们，除了要找到新的工作之外，接下来就是要开始真正接触到实际开发项目，运用所学知识开始在实际项目中实践并最终完成工作任务。或许，你有太多的新技术新知识要学习，但是却不知道该如何着手，不用烦恼，这一切都是学习开始的动力所在。接下来就让我们看看有哪些Spring书籍，可以解决项目问题，助你快速学习成长起来。

## Apress

- Spring Boot 2 Recipes - A Problem-Solution Approach, Apress, 2018
- Beginning Spring Boot 2 Applications and Microservices with the Spring Framework, Apress, 2017
- Spring 5 Recipes - A Problem Solution Approach 4th Edition, Apress, 2017
- Spring Boot Messaging - Messaging APIs for Enterprise and Integration Solutions, Apress, 2017
- Pro Spring Boot, Apress, 2016
- Pro Spring 5th Edition, Apress, 2017
- Pro Spring 4th Edition, Apress, 2014
- Pro Spring Boot, Apress, 2016
- Spring Persistence with Hibernate 2nd Edition, Apress, 2016
- Spring REST, Apress, 2015
- Spring Recipes - A Problem Solution Approach 3rd Edition, Apress, 2014
- Spring Recipes - A Problem Solution Approach 2nd Edition, Apress, 2010
- Spring Recipes - A Problem Solution Approach, Apress, 2008
- Spring Persistence - A Running Start, Apress, 2009
- Spring Enterprise Recipes, Apress, 2009
- The Definitive Guide to Spring Web Flow, Apress, 2008
- Building Spring 2 Enterprise Applications, Apress, 2007
- Expert Spring MVC and Web Flow, Apress, 2006

## Manning

- Spring in Action 5th Edition, Manning, 2018
- Spring Microservices in Action, Manning, 2017
- Spring Boot in Action, Manning, 2016
- Spring in Action 4th Edition, Manning, 2014
- Spring in Practice, Manning, 2013
- Spring Integregation in Action, Manning, 2012
- Spring Roo in Action, Manning, 2012
- Spring Batch in Action, Manning, 2012
- Spring Dynamic Modules in Action, Manning, 2011

- Spring in Action 2nd Edition, Manning, 2007
- iBATIS in Action, Manning, 2007
- Java Persistence with Hibernate, Manning, 2006
- Hibernate In Action, Manning, 2005

## Packt Publishing

- Building RESTful Web Services with Spring 5, 2nd Edition, Packt Publishing, 2018
- Software Architecture with Spring 5.0, Packt Publishing, 2018
- Spring 5.0 By Example, Packt Publishing, 2018
- Spring Boot 2.0 Cookbook, 2nd Edition, Packt Publishing, 2018
- Building Web Apps with Spring 5 and Angular, Packt Publishing, 2017
- Learning Spring 5.0, Packt Publishing, 2017
- Mastering Spring 5.0, Packt Publishing, 2017
- Spring 5 Design Patterns, Packt Publishing, 2017
- Spring 5.0 Cookbook - Recipes to build, test, and run Spring applications efficiently, Packt Publishing, 2017
- Spring 5.0 Microservices, 2nd Edition, Packt Publishing, 2017
- Spring: Developing Java Applications for the Enterprise, Packt Publishing, 2017
- Spring Microservices - Build Scalable Microservices with Spring, Docker, and Mesos, Packt Publishing, 2016
- Spring Security Essentials, Packt Publishing, 2016
- Learning Spring Application Development, Packt Publishing, 2015
- Mastering Spring Application Development, Packt Publishing, 2015
- Mastering Spring MVC 4, Packt Publishing, 2015
- Spring Boot Cookbook, Packt Publishing, 2015
- Spring Batch Essentials, Packt Publishing, 2015
- Spring Cookbook, Packt Publishing, 2015
- Learning Spring Boot, Packt Publishing, 2014
- Spring MVC - Beginner's Guide, Packt Publishing, 2014
- Spring Security 3.1, Packt Publishing, 2012
- Spring Security 3, Packt Publishing, 2010
- Spring Roo 1.1 Cookbook, Packt Publishing, 2011

## O' Reilly Media

- Cloud Native Java - Designing Resilient Systems with Spring Boot, Spring Cloud, and Cloud Foundry, O' Reilly, 2017
- Just Spring, O' Reilly Media, 2011
- Just Spring Integration, O' Reilly Media, 2012

- Spring Data, O'Reilly Media, 2012
- Spring: A Developer's Notebook, O'Reilly Media, 2005

## 国内出版

- Spring 微服务实战, 人民邮电出版社, 2018
- Spring Cloud 微服务实战, 电子工业出版社, 2017
- Spring Boot 实战, 人民邮电出版社, 2016
- Spring Boot 揭秘 - 快速构建微服务体系, 机械出版社, 2016
- Spring 实战 (第4版), 人民邮电出版社, 2016
- Spring技术内幕：深入解析Spring架构与设计原理（第2版），机械工业出版社，2012

以上列出的书单，并不是说要求全部看完，可以根据自己的学习进度和时间安排来选择适合自己的书籍。根据我个人的经验，我在学习一门新技术之前，首先会了解下这门技术有关已经出版的书籍，这能从总体上了解学习的大致范围。拿到一本书先从它的提纲看起，这样就能判断出大体上这本技术包括哪些知识点，以及需要达到什么程度。可以根据自身情况，选择其中学习的优先级和侧重点。有些技术点也许目前还没有用到，可以大致浏览，待以后工作用到再回头学习会更有效果，毕竟理论知识最终要运用于实践中。虽然时间和精力有限，我们也仍然建议选择一两本好书精读，遇到不懂的相关其他技术，可以找一些相关书籍来补充。举例来说，Spring 技术涉及到很多Java EE 规范 (JSP、Servlet、JPA、JMS、Web Service 等)，或许在书中并不会详细介绍这些知识点，那就需要一些更具体的书籍来补充，如Java、Tomcat、Hibernate 等相关书籍。

因此，对于首次接触到Spring技术的同学，我们还为你收集整理了下列有关技术学习的书单：

- Java EE 7 权威指南：卷1和2，机械工业出版社，2015
- Java in a Nutshell, 6th Edition, O'Reilly Media, 2014
- Java Cookbook, 3rd Edition, O'Reilly Media, 2014
- Java 8 in Action - Lambdas, streams, and functional-style programming, Manning, 2014
- JUnit in Action, Second Edition, Manning, 2010
- Tomcat: The Definitive Guide, Second Edition, O'Reilly Media, 2009
- Art of Java Web Development - Struts, Tapestry, Commons, Velocity, JUnit, Axis, Cocoon, InternetBeans, Webwork, Manning, 2006
- Thinking in Java 4th Edition, Prentice Hall, 2006
- Core Servlets and JavaServer Pages: Volume 1: Core Technologies, 2nd Edition, Prentice Hall, 2003
- Expert One-on-One J2EE Development without EJB, Wrox, 2004
- Expert One-on-One J2EE Design and Development, Wrox, 2002
- Test Driven Development, Addison-Wesley Professional, 2002

## 小结

总的来说，Manning出版社的 **in Action** 实战系列的作者一般都是该领域的技术专家，内容深入浅出，技术核心知识讲解透彻。**O'Reilly** (中文：奥莱利) 出版的书籍范围涵盖非常全面，而且也都是很前沿新技术。**O'Reilly** 为软件开发人员带来革命性的“动物书”，相信很多开发人员都读过。除了书籍出版，目前还有在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

Apress 出版社，则是 Manning 出版社有力的补充，技术书籍也相对更新很快。它的“Recipes”与 Packt 出版社的“Cookbook”都是助你成为解决问题专家路上必不可少的“武林秘籍”系列。除此之外，Packt 出版社的在线学习视频课程内容推出很新很快，作者往往聚焦在项目实战上。

写在最后，尽管我一开始想列出所有有关 Spring 的技术书籍（这是我的一点私心），但是总免不了有所遗漏，而且要想满足所有不同层次的读者也不太可能，其中也有部分书籍年代久远（但是并不表示不值得看）。如果你有好的推荐，可以私信告诉我，也可以告诉你的情况，一起来交流学习会更好，我的微信号是：[rainboyan](#)，也可以关注微信公众号/头条号：[Spring 技术社区](#)。

# Spring Boot 最佳实践

Spring Boot 是最流行的用于开发微服务的 Java 框架。在本文中，我将与你分享自 2016 年以来我在专业开发中使用 Spring Boot 所采用的最佳实践。这些内容是基于我的个人经验和一些熟知的 Spring Boot 专家的文章。

在本文中，我将重点介绍 Spring Boot 特有的实践（大多数时候，也适用于 Spring 项目）。如果你想了解 Java 最佳实践，我建议你阅读“Effective Java”，我将在另一篇文章中进行介绍。

以下依次列出了最佳实践，排名不分先后。

## 使用自定义BOM来维护第三方依赖

这条实践是我根据实际项目中的经历总结出的。

Spring Boot 项目本身使用和集成了大量的开源项目，它帮助我们维护了这些第三方依赖。但是也有一部分在实际项目使用中并没有包括进来，这就需要我们在项目中自己维护版本。如果在一个大型的项目，包括了为很多开发模块，那么维护起来就非常的繁琐。

怎么办呢？事实上，[Spring IO Platform](#) 就是做的这个事情，它本身就是 Spring Boot 的子项目，同时维护了其他第三方开源库。我们可以借鉴 [Spring IO Platform](#) 来编写自己的基础项目 [platform-bom](#)，所有的业务模块项目应该以 BOM 的方式引入。这样在升级第三方依赖时，就只需要升级这一个依赖的版本而已。

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>io.spring.platform</groupId>
      <artifactId>platform-bom</artifactId>
      <version>Cairo-SR3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

## 使用自动配置

Spring Boot 的一个主要特性是使用自动配置。这是 Spring Boot 的一部分，它可以简化你的代码并使之工作。当在类路径上检测到特定的 **jar** 文件时，自动配置就会被激活。

使用它的最简单方法是依赖 Spring Boot Starters。因此，如果你想与 Redis 进行集成，你可以首先包括：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

如果你想与MongoDB进行集成，需要这样：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
```

借助于这些 **starters**，这些繁琐的配置就可以很好的集成起来并协同工作，而且它们都是经过测试和验证的。这非常有助于避免可怕的 **Jar 地狱**。

通过使用以下注解属性，可以从自动配置中排除某些配置类：`@EnableAutoConfiguration (exclude = {ClassNotToAutoconfigure.class})`，但只有在绝对必要时才应该这样做。

有关自动配置的官方文档可在 [此处](#)找到。

## 使用 Spring Initializr 来开始一个新的 Spring Boot 项目

这一条最佳实践来自 Josh Long (Spring Advocate, @starbuxman).

Spring Initializr (<https://start.spring.io/>) 为你提供了一个超级简单的方法来创建一个新的 Spring Boot 项目，并根据你的需要来加载可能使用到的依赖。

使用 Initializr 创建应用程序可确保你获得经过测试和验证的依赖项，这些依赖项适用于 Spring 自动配置。你甚至可能会发现一些新的集成，但你可能并没有想到这些。

## 考虑为常见的组织问题创建自己的自动配置

这一条也来自 Josh Long (Spring Advocate, @starbuxman) - 这个实践是针对高级用户的。

如果你在一个严重依赖 Spring Boot 的公司或团队中工作，并且有共同的问题需要解决，那么你可以创建自己的自动配置。

这项任务涉及较多工作，因此你需要考虑何时获益是值得投入的。与多个略有不同的定制配置相比，维护单个自动配置更容易。

如果将这个提供 Spring Boot 配置以开源库的形式发布出去，那么将极大地简化数千个用户的配置工作。

## 正确设计代码目录结构

尽管允许你有很大的自由，但是有一些基本规则值得遵守来设计你的源代码结构。

- 避免使用默认包。确保所有内容（包括你的入口点）都位于一个名称很好的包中。这样就可以避免与装配和组件扫描相关的意外情况。

- 将 `Application.java`（应用的入口类）保留在顶级源代码目录中。
- 我建议将控制器和服务放在以功能为导向的模块中，但这是可选的。一些非常好的开发人员建议将所有控制器放在一起。不论怎样，坚持一种风格！

## 保持 @Controller 的简洁和专注

`Controller` 应该非常简单。你可以在此处阅读 [有关 GRASP 中有关控制器模式部分的说明](#)。你希望控制器作为协调和委派的角色，而不是执行实际的业务逻辑。以下是主要做法：

- 控制器应该是无状态的！默认情况下，控制器是单例，并且任何状态都可能导致大量问题。
- 控制器不应该执行业务逻辑，而是依赖委托。
- 控制器应该处理应用程序的 HTTP 层，这不应该传递给服务。
- 控制器应该围绕用例/业务能力来设计。

要深入这个内容，需要进一步的了解设计 REST API 的最佳实践。无论你是否想要使用 Spring Boot，那都是值得学习的。

## 围绕业务功能构建 @Service

`Service` 是 Spring Boot 的另一个核心概念。我发现最好围绕业务功能/领域/用例（无论你怎么称呼都行）来构建服务。

在应用中设计名称类似 `AccountService`、`UserService`、`PaymentService` 这样的服务，比起像 `DatabaseService`、`ValidationService`、`CalculationService` 这样的会更合适一些。

你可以决定使用 `Controller` 和 `Service` 之间的一对一映射，那将是理想的情况。但这并不意味着，`Service` 之间不能互相调用！

## 使数据库独立于核心业务逻辑之外

我之前还不确定如何在 Spring Boot 中最好地处理数据库交互。在阅读了罗伯特·C·马丁的“Clear Architecture”之后，对我来说就清晰多了。

你希望你的数据库逻辑于服务分离出来。理想情况下，你不希望服务知道它正在与哪个数据库通信，这需要一些抽象来封装对象的持久性。

罗伯特C.马丁强烈地说明，你的数据库是一个“细节”。这意味着你的应用程序将不与特定数据库耦合。过去很少有人会切换数据库。我注意到，使用 Spring Boot 和现代微服务开发 - 一些事情将变得更快。

## 保持业务逻辑不受 Spring Boot 代码的影响

考虑到“Clear Architecture”的教训，你还应该保护你的业务逻辑。将各种 Spring Boot 代码混合在一起是非常诱人的……不要这样做。如果你能抵制诱惑，你将保持你的业务逻辑可重用。

部分服务通常成为库。如果你不必从代码中删除大量 Spring 注解，则更容易创建。

## 推荐使用构造函数注入

这一条实践来自 Phil Webb (Spring Boot 的项目负责人, `@phillip_webb`)。

保持业务逻辑免受 Spring Boot 代码侵入的一种方法是使用构造函数注入。不仅是因为 `@Autowired`

注解在构造函数上是可选的，而且还可以在没有 Spring 的情况下轻松实例化 bean。

## 熟悉并发模型

我写过的最受欢迎的文章之一是 “[介绍 Spring Boot 中的并发](#)”。我认为这样做的原因是这个领域经常被误解和忽视。如果使用不当，就会出现问题。

在 Spring Boot 中，[Controller](#) 和 [Service](#) 是默认是单例。如果你不小心，这会引入可能的并发问题。你通常也在处理有限的线程池。请熟悉这些概念。

如果你正在使用新的 [WebFlux](#) 风格的 Spring Boot 应用程序，我已经解释了它在 “[Spring’s WebFlux/Reactor Parallelism and Backpressure](#)” 中是如何工作的。

## 加强配置管理的外部化

这一点超出了 Spring Boot，虽然这是人们开始创建多个类似服务时常见的问题……

你可以手动处理 Spring 应用程序的配置。如果你正在处理多个 Spring Boot 应用程序，则需要使配置管理能力更加强大。

我推荐两种主要方法：

- 使用配置服务器，例如 [Spring Cloud Config](#)
- 将所有配置存储在环境变量中（可以基于 git 仓库进行配置）

这些选项中的任何一个（第二个选项多一些）都要求你在 DevOps 更少工作量，但这在微服务领域是很常见的。

## 提供全局异常处理

你真的需要一种处理异常的一致方法。Spring Boot 提供了两种主要方法：

- 你应该使用 [HandlerExceptionResolver](#) 定义全局异常处理策略
- 你也可以在控制器上添加 [@ExceptionHandler](#) 注解，这在某些特定场景下使用可能会很有用

这与 Spring 中的几乎相同，并且 Baeldung 有一篇关于 [REST 与 Spring 的错误处理](#) 的详细文章，非常值得一读。

## 使用日志框架

你可能已经意识到这一点，但你应该使用 [Logger](#) 进行日志记录，而不是使用 [System.out.println\(\)](#) 手动执行。这很容易在 Spring Boot 中完成，几乎没有配置。只需获取该类的记录器实例：

```
Logger logger = LoggerFactory.getLogger(MyClass.class);
```

这很重要，因为它可以让你根据需要设置不同的日志记录级别。

## 测试你的代码

这不是 Spring Boot 特有的，但它需要提醒——测试你的代码！如果你没有编写测试，那么你将从一开始就编写遗留代码。

如果有其他人使用你的代码库，那边改变任何东西将会变得危险。当你有多个服务相互依赖时，这甚至可能更具风险。

由于存在 Spring Boot 最佳实践，因此你应该考虑将 Spring Cloud Contract 用于你的消费者驱动契约 [2: Consumer-Driven Contracts: A Service Evolution Pattern]。它将使你与其他服务的集成更容易使用。

## 使用测试切片让测试更容易，并且更专注

这一条实践来自 Madhura Bhave (Spring 开发者, [@madhurabhave23](#))。

使用 Spring Boot 测试代码可能很棘手 -  
你需要初始化数据层，连接大量服务，模拟事物……实际上并不是那么难！答案是 - 使用测试切片。

使用测试切片，你可以根据需要仅连接部分应用程序。这可以为你节省大量时间，并确保你的测试不会与未使用的内容相关联。来自 [spring.io](#) 的一篇名为 [Custom test slice with Spring test 1.4](#) 的博客文章解释了这种技术。

## 总结

感谢 Spring Boot，编写基于 Spring 的微服务变得前所未有的简单。我希望通过这些最佳实践，你的实施过程不仅会很快，而且从长远来看也会更加强大和成功。祝你好运！

## 感谢！

我要感谢以下人员帮助我改进这篇文章：

- Marcin Grzejszczak ([@MGrzejszczak](#)) – 转发我的博文并引起 Spring 团队的注意
- Josh Long ([@starbuxman](#)) – 给我反馈并添加了其他最佳实践
- Phil Webb ([@phillip\\_webb](#)) – 给我反馈并添加了其他最佳实践
- Madhura Bhave ([@madhurabhave23](#)) – 给我反馈并添加了其他最佳实践

非常感谢以上的那些哥们儿！Spring Boot 有一个非常棒的社区！

声明：除了第一条实践来自我本人，其他的都是原文内容。如有转载，请注明本文出处。

原文：[Spring Boot – Best Practices](#)  
作者：[Bartosz Jedrzejewski](#)  
翻译：[Rain](#)

# 10 种保护Spring Boot应用程序的绝佳方法

Spring Boot 大大简化了 Spring 应用程序的开发。它的自动配置和起步依赖减少了启动应用程序所需的代码和配置量。如果你之前在 Spring 中使用过大量的 XML 配置，那么你会觉得 Spring Boot 就是一股清新的空气。

Spring Boot 于2014年首次发布，自那以后更新了很多。就像代码质量和测试一样，安全性已成为开发人员非常关心的问题。如果你是一名开发人员却对安全并不关心，那么你应该考虑下是否应该如此。本文旨在教你如何创建更安全的 Spring Boot 应用程序。

我与 [Simon Maple](#) 合作完成了这篇文章，他是一名 Java 冠军程序员和 Snyk 的开发者关系总监。我们都在安全行业的公司工作，热爱 Java，并希望帮助开发人员创建更安全的应用程序。我们认为以撰写这篇文章这种方式来回馈社区是有趣的。除了下面这些我们已列出的，如果你还有其他的建议，请在评论中告诉我们！

## 在生产中使用 HTTPS

传输层安全性（TLS）是 HTTPS 的官方名称。你可能听说过它称为 SSL（安全套接字层）。SSL 是已弃用的名称，TLS 是一种加密协议，可通过计算机网络提供安全通信。其主要目标是确保计算机应用程序之间的隐私和数据完整性。

过去，TLS/SSL 证书很昂贵，而且 HTTPS 被认为很慢。机器的运算速度极大提升解决了性能问题，而 [Let's Encrypt](#) 提供免费的 TLS 证书。这两项发展改变了现状，并使 TLS 很快成为主流。

2018年7月24日，Google Chrome 已将 [HTTP 网站标记为“不安全”](#)。虽然这在网络社区引起了相当多的争议，但它已然如此。知名安全研究员 Troy Hunt 创建了一个 [“为什么不使用 HTTPS?”](#) 来记录不使用 HTTPS 的大型网站。你可能没有开发下一个大型网站，但为什么限制自己不使用呢？

生成和更新加密 TLS 证书可以自动化。由于它们是免费的，所以没有理由不去做！[Spring Boot Secured By Let's Encrypt](#) 是介绍如何做到这一点的有用指南。

要在 Spring Boot 应用程序中强制使用HTTPS，你可以扩展 [WebSecurityConfigurerAdapter](#) 并要求安全连接。

```
@Configuration
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.requiresChannel().requiresSecure();
    }
}
```

这个配置会强制在开发时也使用HTTPS，这可能会很麻烦，因为你必须使用自签名证书。如果你正在使用 Heroku，Cloud Foundry 或其他云提供商，更合理的配置是寻找 [X-Forwarded-Proto](#) 请求头。

```

@Configuration
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.requiresChannel()
            .requestMatchers(r -> r.getHeader("X-Forwarded-Proto") != null)
            .requiresSecure();
    }
}

```

云提供商可以大大简化 TLS 证书。[亚马逊证书管理器](#)与 Let's Encrypt 完全相同，但默认情况下它内置于所有 AWS 产品或服务中。它允许你提供 100% 免费的 SSL 证书并处理自动续订等，只需零配置工作。Heroku 也有自动证书管理。

另一个重要的事情是使用HTTP严格传输安全性（HSTS）。HSTS 是一种 Web 安全策略机制，可以保护网站免受协议降级攻击和 Cookie 劫持。服务器使用名为 [Strict-Transport-Security](#) 的响应头字段将HSTS策略传送到浏览器。Spring Security 默认发送此标头，以避免在开始时出现不必要的 HTTP 跃点。

## 借助 Snyk 检查你的依赖（Check Your Dependencies with Snyk）

你很可能不知道应用程序使用了多少直接依赖项，你也很可能不知道应用程序使用了多少传递依赖项。这通常是正常的，尽管依赖性构成了整个应用程序的大部分。攻击者越来越多地针对开源依赖项，因为它们的重用为恶意黑客提供了许多受害者。确保应用程序的整个依赖关系树中没有已知的漏洞非常重要。

**Snyk**  
测试你的应用程序构建工件，标记那些已知漏洞的依赖项。它在仪表板为你提供了在应用程序中使用的软件包中存在的漏洞列表。

## All vulnerable projects

[See all projects](#)**M sjmaple/java-goof:todolist-web-struts/pom.xml****17 H** **22 M** **2 L** Updated a day agoDependencies: 47 • Source: [GitHub](#)**M sjmaple/spring.goof:pom.xml****6 H** **5 M** **0 L** Updated a day agoDependencies: 61 • Source: [GitHub](#)**M sjmaple/java-goof:todolist-web-common/pom.xml****3 H** **4 M** **0 L** Updated 8 hours agoDependencies: 34 • Source: [GitHub](#)

此外，它还将建议升级版本或提供补丁以通过针对源代码存储库的拉取请求来修复你的安全问题。Snyk还保护你的环境，在你的存储库上提交的任何拉取请求都将通过Webhooks触发自动测试，这样可以确保它们不会引入新的已知漏洞。

每天都会在现有项目和库中发现新的漏洞，因此监控和保护生产部署也很重要。Snyk创建快照并监控你的部署，以便在发现新漏洞时，你可以通过喜欢的渠道、JIRA、Slack或电子邮件自动接收通知，并创建拉取请求以提供新漏洞的升级和补丁。

Snyk可通过Web界面和命令行界面获得，因此你可以轻松地将其与持续集成环境集成，并将其配置为当存在严重性超出设定阈值的漏洞时中断构建。

你可以[免费使用 Snyk](#)用于开源项目，也可以用于私有项目，但每月只有有限次数的测试。

## 升级到最新版本

定期升级应用程序中的依赖项有多种理由，安全性是让你有升级动力的最重要原因之一。[start.spring.io](#)启动页面尽可能使用最新版本的Spring软件包以及依赖项。

我发现在依赖项中寻找漏洞可能有助于激励人们升级。然而，有大量证据表明并非所有CVE都被报道。一般来说，我发现理想（也许不实用）的解决方案是最新和最好的。

- 

— Rob Winch

基础架构升级通常不如依赖项升级具有破坏性，因为库的作者对向后兼容性和版本之间的行为更改的敏感性

各不相同。话虽如此，当你在配置中发现安全漏洞时，你有三种选择：升级、修补程序或忽略。

在对应用程序进行必要的更改以使用较新版本之后，就应用程序的整体运行状况而言，升级是最安全的。

对易受攻击的项目的修补程序消除了程序包中的漏洞，但通常会留下一个可能未经过充分测试的配置。由于修补程序只会更改易受攻击的代码，因此对库的代码更改将会减少，同时可以降低破坏向后兼容性或引入行为更改的可能性。像Snyk这样的第三方安全公司为许多漏洞提供了补丁程序，因此如果无法升级到更新版本的库，你仍然可以使用旧版本的补丁程序。

当然，忽略漏洞是一种选择，但不是一个好选择。也许你知道一个漏洞，但不相信它是可以直接利用的。请记住，它可能不在你的应用程序流程中，但在某些时候，开发人员可能会添加使用易受攻击路径的其他代码。

## 启用 CSRF 保护

**跨站点请求伪造**是一种攻击，强制用户在他们当前登录的应用程序中执行不需要的操作。如果用户是普通用户，则成功攻击可能涉及状态更改请求，如转移资金或更改其电子邮件地址。如果用户具有提升的权限，则CSRF攻击可能会危及整个应用程序。

Spring Security 具有出色的 CSRF 支持，默认情况下处于启用状态。如果你正在使用 Spring MVC 的`<form:form>`标签或 Thymeleaf 并且添加了注解 `@EnableWebSecurity`，则 CSRF 标记将作为隐藏输入字段自动添加。

如果你正在使用像 Angular 或 React 这样的 JavaScript 框架，则需要配置 `CookieCsrfTokenRepository`，以便 JavaScript 可以读取 cookie。

```
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .csrf()
            .csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse());
    }
}
```

如果你使用的是 Angular，这样设置就够了。如果你使用的是 React，则需要从 cookie 中读取 `XSRF-TOKEN` 的值并作为标头 `X-XSRF-TOKEN` 发回。

当请求是 HTTPS 协议时，Spring Security 会自动为 `XSRF-TOKEN` cookie 添加安全标志。Spring Security 不对 CSRF cookie 使用 `SameSite = strict` 标志，但在使用 Spring Session 或 WebFlux 会话会这样处理。会话 cookie 对于识别用户是有意义，但它对 CSRF cookie 没有多少用处，因为 CSRF 令牌也需要存放在请求中。

## 使用内容安全策略来防止 XSS 攻击

内容安全策略（Content Security Policy，简称“CSP”）是一个增加的安全层，可帮助缓解 XSS（跨站点脚本）和数据注入攻击。要启用它，你需要配置应用程序以返回 `Content-Security-Policy` 标头。你还可以在 HTML 页面中使用 `<meta http-equiv="Content-Security-Policy">` 标记。

Spring Security 默认提供了许多安全标头：

Cache-Control: no-cache, no-store, max-age=0, must-revalidate  
 Pragma: no-cache  
 Expires: 0  
 X-Content-Type-Options: nosniff  
 Strict-Transport-Security: max-age=31536000 ; includeSubDomains  
 X-Frame-Options: DENY  
 X-XSS-Protection: 1; mode=block

Spring Security 默认不添加 CSP。你可以使用以下配置在 Spring Boot 应用程序中启用 CSP 标头。

```
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.headers()
            .contentSecurityPolicy("script-src 'self' https://trustedscripts.example.com; object-src
https://trustedplugins.example.com; report-uri /csp-report-endpoint/");
    }
}
```

CSP 是防止 XSS 攻击的良好防御。请记住，打开 CSP 以允许 CDN  
 这也意味着允许访问许多非常古老且易受攻击的 JavaScript 库。这意味着使用 CDN  
 通常对应用程序的安全性没有多大帮助。

你可以使用 [securityheaders.com](https://securityheaders.com) 测试你的 CSP 标头是否与实际是一致的。

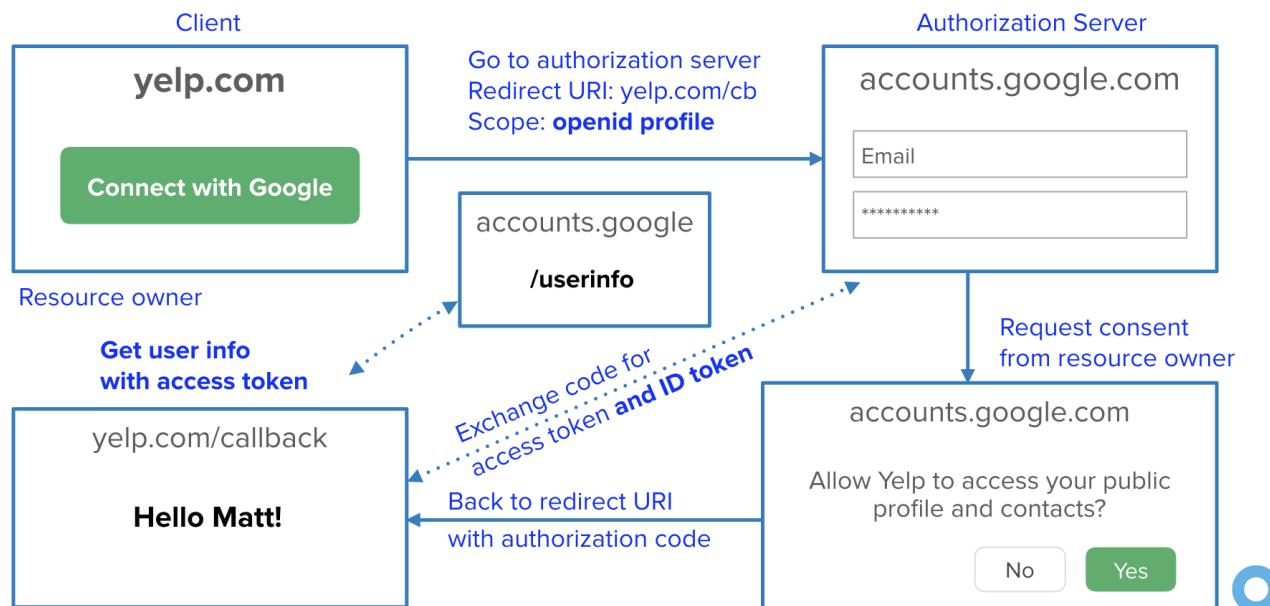
## 使用 OpenID Connect 进行身份验证

OAuth 2.0 是行业标准的授权协议。它使用范围来定义授权用户可以执行的操作的权限。但是，OAuth 2.0  
 不是身份验证协议，并且不提供有关经过身份验证的用户的信息。

OpenID Connect (OIDC) 是一个 OAuth 2.0 扩展，提供用户信息。除了访问令牌之外，它还添加了 ID  
 令牌，以及可以从中获取其他信息的 [/userinfo](#) 端点。它还添加了端点发现功能和动态客户端注册。

下图显示了 OIDC 如何用于身份验证。

# OIDC Authorization Code Flow



如果使用 OIDC

进行身份验证，则无需担心存储用户、密码或对用户进行身份验证。相反，你将使用身份提供商（IdP）为你执行此操作。你的 IdP 甚至可能提供多因素身份验证（MFA）等安全附加组件。

要了解如何在 Spring Boot 应用程序中使用 OIDC，请参阅 [Spring Security 5.0 和 OIDC 入门](#)。要概括如何使用它，你首先需要向项目添加一些依赖项，然后在 `application.yml` 文件中配置一些属性。

```
spring:
  security:
    oauth2:
      client:
        registration:
          okta:
            client-id: {clientId}
            client-secret: {clientSecret}
            scope: openid email profile
        provider:
          okta:
            issuer-uri: https://{{yourOktaDomain}}/oauth2/default
```

NOTE

使用 `issuer-uri` 仅在 Spring Security 5.1 中支持，Spring Security 5.1 正在积极开发中并计划于 2018 年 9 月发布。

你可以使用像 [Keycloak](#) 这样的开源系统来设置自己的 OIDC 服务器。如果你不想在生产中维护自己的服务器，可以使用 Okta 的 Developer

API。立即注册免费帐户，每月在 [developer.okta.com/signup](https://developer.okta.com/signup) 上获得 1000 名活跃用户！

如果你想使用 OAuth 2.0，OIDC 及其允许的不同流程，请参阅 <https://www.oauth.com/playground>。此站点不需要你创建帐户，但它确实使用了 Okta 的开发人员 API。

## 管理密码？ 使用密码哈希！

以纯文本格式存储密码是你可以为应用程序的安全性做的最糟糕的事情之一。幸运的是，Spring Security 默认情况下不允许使用纯文本密码。它还附带了一个加密模块，可用于对称加密、密钥生成和密码散列（也就是密码编码）。

**PasswordEncoder** 是 Spring Security 中密码散列的主要接口，如下所示：

```
public interface PasswordEncoder {
    String encode(String rawPassword);
    boolean matches(String rawPassword, String encodedPassword);
}
```

Spring Security 提供了几种实现，最常用的是 **BCryptPasswordEncoder** 和 **Pbkdf2PasswordEncoder**。

对于一般的密码管理，我们建议使用 SCrypt 或 Argon2。SCrypt 现在已经过时了（已经有一段时间了），并且有一个 BCrypt 没有的额外的复杂因素，这使得暴力成倍地变得更加困难或者昂贵。它由著名的密码学家/安全人员（Colin Percival）编写，并且在几乎所有编程语言中都有很好的库。SCrypt 也得到 Latacora 的认可。

来自 Okta Developer Relations 团队的密码专家 Randall Degges：

### Argon2

相对较新（现在已有几年历史），但已经过广泛的审核，并且是许多组织在几年内参与的加密哈希挑战的结果。毫无疑问，它们的“最强”散列算法都增加了 scrypt 没有的另一层复杂性，这使得与 scrypt 相比它被暴力破解的难度成倍地增加。Argon2 非常棒，我在几种语言中使用它取得了巨大的成功，但如果你担心过于尖锐的 scrypt 是一个安全的赌注，而不是有争议的。

来自 Spring Security 负责人 Rob Winch：

### 我喜欢

BCrypt，但一般的建议是单向自适应哈希。某些用户出于合规性原因可能需要使用 PBKDF2。有一个为 Argon2 支持记录的票证，但是我找不到任何 Apache 2 协议开源的原生 Java 实现（如果你知道，请告诉我们！）。相反，这些库依赖于他们委托的二进制文件，从我的角度来看，这是不理想的。我们处于到底是考虑备选还是最大化利用二进制委托实现的选择矛盾之中。（We are on the fence about waiting vs. leveraging one of the implementations that delegate to a binary.）

对于那些想要使用 SCrypt 的人，可以通过 **SCryptPasswordEncoder** 中的 Bouncy Castle 在 Spring Security 中获得支持。Spring Security 5.1（即2018年9月下旬）将附带 **UserDetailsPasswordService**

API，允许你升级密码存储。

## 安全地存储秘密（Secrets）

应谨慎处理敏感信息，如密码、访问令牌等。你不能将它们随意的以纯文本形式传递，或者为了被访问而将它们保存在本地存储中。由于 GitHub 的历史已经一次又一次证明，开发人员并没有仔细考虑如何存储他们的秘密。

当然，你能够也应该加密你的敏感数据，例如密码。现在你的密码是安全的，你有一个新的秘密：你的解密钥！你对这个新秘密有什么看法？也许存放在本地？也许在另一个地方，你认为攻击者很难找到它。这不能解决问题：它只是暂时被推迟了。如果没有适当的流程，你只是让黑客稍微难以解开你的秘密而已。

一个好的做法是将保密信息存储在保管库中，该保管库可用于存储，还提供对应用程序可能使用的服务的访问权限，甚至生成凭据。HashiCorp的Vault使得存储机密变得轻而易举，并提供了许多额外的服务。可以配置保险柜，以便没有人可以访问所有数据，不提供单点控制。根密钥 Vault 定期被更改，仅存储在内存中。它有一个主开关，当被触发时会密封你的保管库，如果出现问题就阻止它共享秘密。Vault 使用分配给可以限定特定用户，服务或应用程序的策略的令牌。你还可以与 LDAP 等常用身份验证机制集成以获取令牌。

除了没有问题的“黄金路径视图”（golden-path view）外，Vault 还可以提供应用在被黑客入侵时的备选方案。此时，撤销单个或多个秘密很重要，可能是由特定用户或特定类型。Vault 提供了一种自动化的方式，可以在时机成熟时快速完成。

如果你对此感兴趣，请务必花一些时间查看 Spring Vault，它为 HashiCorp Vault 添加抽象，为客户端提供基于 Spring 注解方式，允许访问、存储和撤销机密而不会迷失在底层架构中。以下代码段显示了使用注释从 Spring Vault 中提取密码的难易程度。

```
@Value("${password}")
String password;
```

## 使用 OWASP 的 ZAP 测试你的应用程序

OWASP ZAP 安全工具是一个代理，可在运行时对你的实时应用程序执行渗透测试。它托管在 GitHub 上，是一个受欢迎的（超过4k星）免费的开源项目。

OWASP ZAP 用于查找漏洞的两种方法是 Spider 和 Active Scan。Spider 工具以 URL 种子开头，它将访问并解析每个响应，识别超链接并将它们添加到列表中。然后，它将这些新找到的 URL 并以递归方式继续遍历，为你的Web应用程序创建 URL 映射。Active Scan 工具将根据潜在漏洞列表自动测试你选择的目标。它为你提供了一个报告，显示你的 Web 应用程序可被利用的地方，以及有关漏洞的详细信息。

## 让你的安全团队进行代码审查

代码评审对任何高效的软件开发团队都至关重要。在 Okta，我们所有的生产代码和官方开源项目都需要通过我们的安全专家团队进行分析。你的公司可能没有安全专家，但如果你正在处理敏感数据，也许你应该这样做！

## 不要因为你的缺乏安全感而令人不安

Okta有一些很棒的T恤，上面写着“我发现你缺乏安全感”令人不安。我们喜欢在机场旅行时听到戴上乳胶手套的声音。不要做一名缺乏安全性的 Spring Boot 应用程序开发人员！



原文：[10 Excellent Ways to Secure Your Spring Boot Application](#)  
作者：Matt Raible  
翻译：Rain

# 介绍 Spring Cloud Function 项目

Spring Cloud Function 是一个新项目，具有以下高阶目标：

- 通过函数促进业务逻辑的实现
- 将业务逻辑的开发生命周期与任何特定运行时目标分离，以便相同的代码可以作为Web端点，流处理器或任务运行
- 支持无服务器提供商之间的统一编程模型，以及独立运行（本地或PaaS）的能力
- 在无服务器提供商上启用Spring Boot功能（自动配置，依赖注入，指标）

就像Spring一直在推广基于普通java对象（POJO）的编程模型一样，Spring Cloud Function也基于普通的函数来推广编程模型。我们指的是 `java.util.function` 包中定义的核心接口：`Function`, `Consumer` 和 `Supplier`。

这些类型的实现可以通过 `@FunctionScan` 启用的类路径扫描显式或隐式地注册为 Bean。参数和（或）返回类型可以选择使用 Reactor 的 `Flux`，它是 Reactive Streams 的 `Publisher`。这样可以实现与其他 Reactive Streams 组件的互操作性，甚至是基于其他实现的组件，例如 RxJava 2，并且它为此处理模型带来了非阻塞IO和反压等反应性功能（有关更多信息，请参阅 Project Reactor）。只要参数和（或）返回类型不是 `Flux`，Spring Cloud Function 就会将它们包装起来，以便函数可以通过 `Flux` 进行互操作。对于简单的一次性处理用例，您可以保持简单：

```
public class Greeter implements Function<String, String> {
    public String apply(String name) {
        return "Hello " + name;
    }
}
```

但是，如果您需要通过窗口函数或 `reduce` 方法来实现处理数据集作为处理单元的函数，则可以使用 `Flux` 类型：

```
public static class WordCount
    implements Function<Flux<String>, Flux<Map<String, Integer>>> {
    public Flux<Map<String, Integer>> apply(Flux<String> phrases) {
        return phrases.window(3)
            .flatMap(f -> f.flatMap(phrase -> Flux.fromArray(phrase.split("\\W"))))
            .reduce(new HashMap<String, Integer>(), (map, word) -> { map.merge(word, 1, Integer::sum); return map; }));
    }
}
```

依靠函数类型也可以轻松编写功能，例如：

```
twistAndShout = twist.andThen(shout);
```

当然，也可以使用lambdas定义函数，例如：

```
Function<String, String> shout = s -> s.toUpperCase() + "!";
```

实际上，Spring Cloud Function 支持动态地将基于 String 的 lambdas 编译为函数实例。这在原型设计或添加一些简单的转换逻辑时尤其有用，因为今天常用的是Spring表达式语言。

您可能会问为什么 Spring 需要推广此模型，因为它无论如何都可以轻松创建 **Function**, **Consumer** 和 **Supplier**

实例。要知道答案涉及控制反转，这应该不足为奇。多年来，好莱坞原则已经描述了从基本依赖注入到 Spring 无处不在地使用模板模式的所有内容：“不要打电话给我们，我们会打电话给你”。上面提到的 **Flux-adapting**

实际上是控制反转的一个例子，但更重要的是业务逻辑与部署配置文件的分离。在这种情况下，业务逻辑指的是函数，而部署配置文件可以是 REST 应用程序，流处理应用程序或有限任务。Spring Cloud Function 为每种类型提供 JAR，并且在每种情况下，自动配置的 **FunctionCatalog** 用于在 **ApplicationContext** 中定位 **Functions**, **Consumers** 和 **Suppliers**。

例如，要将上面显示的 Greeter 函数部署为 REST 端点，只需要添加 “spring-cloud-function-web” 依赖关系，如此 **POM** 中所示。这还包括 Spring Boot Maven 插件，以便构建生成可执行的 JAR：

```
./mvnw clean install  
java -jar greeter/target/greeter-0.0.1-SNAPSHOT.jar
```

然后可以使用curl调用它：

```
$ curl -H "Content-Type: text/plain" :8080/greeter -d World  
Hello World
```

同样，要将功能部署为流处理器（stream-processor），只需要添加 “spring-cloud-function-stream” 依赖性，而这种依赖性依赖于 Spring Cloud Stream。正如 Spring Cloud Stream 提供了 **Binder抽象**，无需定义通道适配器。Spring Cloud Function 无需声明 Service Cloudators, Transformers 等组件，甚至是 Spring Cloud Stream 委派的使用了 **@StreamListener** 注解的方法。“spring-cloud-function-stream” JAR 本身提供了所有这些功能。这是另一个将反转控制提升到另一个层次的案例。

在本博客系列的第 2 部分中，我们将提供在下一版 Spring Cloud Data Flow 中如何使用 **Suppliers**, **Functions** 和 **Consumers**

的示例。基本思想是，无论何时需要提供一些自定义逻辑，您都可以实现简单的函数。这是一个可选择模型的完美例子，你不仅不需要提供样板，而且最好还是让框架处理它。

### **mySupplier | myFunction | myConsumer**

部署配置文件甚至可以扩展到无服务器（即，函数即服务）提供商，例如 AWS Lambda 和 Apache OpenWhisk（以及 Azure 函数和 Google 云函数，一旦它们提供对 Java 的支持）。在本博客系列的第 3 部分中，我们将深入了解该主题的更多详细信息，但是现在您可以仔细阅读 **AWS Lambda 适配器** 和 **Apache OpenWhisk 适配器** 的文档。即将发布的博客还将介绍与基于 Kubernetes 的无服务器框架（如 Fission）的集成。

除了将业务逻辑和基础架构分离之外，各种部署配置文件 JAR 和 FaaS 适配器都可以提高可移植性。开发人员可以完全隔离实现一个函数，包括仅关注输入和输出参数的单元测试。然后，该函数可以与依赖包一起打包，该依赖包允许它在目标环境中运行，从独立的 REST 应用程序到 Spring

Cloud Data Flow 或 FaaS 提供商都能支持。

这将我们带到这个介绍性博客的最后一点。“Serverless”一词产生了很多激烈争论，几乎总是会有解释：“当然还有服务器，但你不必考虑它们。”因此，虽然我们将拒绝引入“无框架”，相同的概念确实可以应用于框架。在上面的 Spring Cloud Data Flow 示例中，函数开发人员不需要考虑框架，甚至不需要生成在其依赖项中具有任何框架代码的工件。同样的想法适用于 FaaS

适配器。我们基本上把控制反转推到了能够将好莱坞原则转变为：“不依赖于我们，我们将依靠你”的程度。这在好莱坞可能不会很好，但对于开发人员而言，这意味着您只需编写一个函数，将其打包到 JAR 中，并将其注册以用于各种端点或适配器。一如既往，Spring 遵循 Alan Kay

强力主张的原则：“简单的事情应该很简单，复杂的事情应该是可能的。”

在即将发布的博客文章中，我们将深入探讨 Spring Cloud Function 带来的一些更复杂的特性，但我们永远不会忘记保持事情简单。

NOTE

原文：<https://spring.io/blog/2017/07/05/introducing-spring-cloud-function>

翻译：春之雨

说明：版权归原作者所有，翻译仅供学习参考，勿做商用。时间仓促，翻译不免出错，欢迎反馈和讨论，感谢阅读。

# 介绍 Spring Data JDBC

Spring Data Lovelace版本即将发布，本文将介绍其中的Spring Data新模块：**Spring Data JDBC**。

## Spring Data JDBC

背后的思想是提供对关系数据库的访问，而无需服从JPA的复杂性。JPA提供延迟加载，缓存和脏跟踪等功能。这些特性当然都很棒，如果你需要这些它们，可以考虑JPA，但是实际使用起来比以想象的要难。

延迟加载可能会在你不期望它时触发昂贵的操作，或者它可能会因异常而失败。当你真正想要比较一个被修改过的实体的两个版本时，缓存可能会妨碍你很难顺利处理所有持久性操作。

## Spring Data JDBC

旨在实现更简单的模型。不会有缓存，脏跟踪或延迟加载。相反，只有在调用存储库方法时才会发出SQL语句。作为该方法返回的对象在方法返回之前完全加载。实体没有“会话”和代理。所有这些都应该使Spring Data JDBC更容易理解。

当然，这种更简单的方法会导致约束，这些约束将在未来的帖子中介绍。此外，它是第一个版本，尽管我们想要并计划实施许多功能，但我们不得不推迟以便你尽早使用到这些内容。

我们来看一个简单的例子

```
class Customer {
    @Id
    Long id;
    String firstName;
    LocalDate dob;
}
```

请注意，你不需要getter或setter。如果你愿意，也可以使用它们。实际上，唯一的要求是实体有一个用Id注释的属性（这个是 [@org.springframework.data.annotation.Id](#)，而不是 [javax.persistence](#)）。

接下来，我们需要声明一个repository，最简单的方式是让它扩展 [CrudRepository](#)。

```
interface CustomerRepository extends CrudRepository<Customer, Long> {}
```

最后，我们需要配置ApplicationContext以启用repositories的创建。

```

@Configuration
@EnableJdbcRepositories (1)
public class CustomerConfig extends JdbcConfiguration { (2)

    @Bean
    NamedParameterJdbcOperations operations() { (3)
        return new NamedParameterJdbcTemplate(dataSource());
    }

    @Bean
    PlatformTransactionManager transactionManager() { (4)
        return new DataSourceTransactionManager(dataSource());
    }

    @Bean
    DataSource dataSource(){ (5)
        return new EmbeddedDatabaseBuilder()
            .generateUniqueName(true)
            .setType(EmbeddedDatabaseType.HSQL)
            .addScript("create-customer-schema.sql")
            .build();
    }
}

```

让我们一步一步地完成配置。

1. [EnableJdbcRepositories](#) 可以创建存储库。由于它需要存在一些bean，我们需要其余的配置。

2. 扩展 [JdbcConfiguration](#) 会向 ApplicationContext 添加一些默认 bean。你可以覆盖其方法以自定义 Spring Data JDBC 的某些行为。现在，我们使用默认实现。

3. 真正重要的部分是 [NamedParameterJdbcOperations](#)，它在内部用于向数据库提交 SQL 语句。

4. 严格来说，事务管理不是必需的。但是你的工作不支持跨越多个操作的交易，没有人想要，对吧？

5. Spring Data JDBC 不直接使用 [DataSource](#)，但是，由于 [TransactionManager](#) 和 [NamedParameterJdbcOperations](#) 需要它，因此将其注册为 bean 是确保两者使用相同实例的简单方法。

这就是你开始使用它所需要的一切。现在让我们在测试中运行它：

```

@RunWith(SpringRunner.class)
@Transactional
@ContextConfiguration(classes = CustomerConfig.class)
public class CustomerRepositoryTest {

    @Autowired CustomerRepository customerRepo;

    @Test
    public void createSimpleCustomer() {

        Customer customer = new Customer();
        customer.dob = LocalDate.of(1904, 5, 14);
        customer.firstName = "Albert";

        Customer saved = customerRepo.save(customer);

        assertThat(saved.id).isNotNull();

        saved.firstName = "Hans Albert";

        customerRepo.save(saved);

        Optional<Customer> reloaded = customerRepo.findById(saved.id);

        assertThat(reloaded.isPresent());

        assertThat(reloaded.get().firstName).isEqualTo("Hans Albert");
    }
}

```

## @Query 注解

你可能不会仅仅使用 [CrudRepository](#) 中的基本 CRUD 方法。在 Spring Data 后续版本中，有个更强大的特性是通过方法名称来推导查询，但是我们决定推迟查询推导。在这之前，你可以使用简单的 [@Query](#) 注解来指定存储库查询方法：

```

@Query("select id, first_name, dob from customer where upper(first_name) like '%' || 
upper(:name) || '%' ")
List<Customer> findByName(@Param("name") String name);

```

请注意，如果使用 **-parameters** 标志进行编译，则不需要 **@Param** 注解。

如果要执行更新或删除操作，则可以向方法添加 **@Modifying** 注解。

让我们创建另一个测试以试用新方法。

```
@Test
public void findByName() {

    Customer customer = new Customer();
    customer.dob = LocalDate.of(1904, 5, 14);
    customer.firstName = "Albert";

    Customer saved = customerRepo.save(customer);

    assertThat(saved.id).isNotNull();

    customer.id= null; (1)
    customer.firstName = "Bertram";

    customerRepo.save(customer);

    customer.id= null;
    customer.firstName = "Beth";

    customerRepo.save(customer);

    assertThat(customerRepo.findByName("bert")).hasSize(2); (2)
}
```

1.由于Java对象与其对应行之间的连接只是其 **Id** 加上其类型，因此将 **Id** 设置为 **null** 并再次保存它会在数据库中创建一条新记录。

2.我们正在进行不区分大小写 (like) 搜索，因此，我们找到 “Albert” 和 “Bertram”，但不是 “Beth”

## 小结

你可以查看[示例](#), [文档](#), 当然还有[源代码](#)。如果你有任何疑问, 请在[StackOverflow](#)上提问。如果你发现错误或想要请求新功能, 请[创建一个问题](#)。

NOTE

原文：<https://spring.io/blog/2018/09/17/introducing-spring-data-jdbc>  
 编译：春之雨  
 说明：版权归原作者所有，翻译仅供学习参考。欢迎反馈和讨论，感谢阅读。

Spring in 2018

# Spring Cloud Function 对 Kotlin 语言的支持

我们很少发表关于单个功能的博客，但考虑到这个功能是 Spring Cloud Function（相对年轻的项目）中最受欢迎的功能之一，我们认为它可能是合适的，所以才写了这篇文章。

Spring Cloud Function 已经添加了对 Kotlin lambdas 的初始支持。这意味着 Spring Cloud Function 现在可以识别出与 Java 的 Provider, Function 或 Consumer 之一有效匹配的 Kotlin lambdas，并将其视为这样。

这样，

```
@Bean  
open fun kotlinFunction(): (String) -> String {  
    return { it.toUpperCase() }  
}  
  
@Bean  
open fun kotlinConsumer(): (String) -> Unit {  
    return { println(it) }  
}  
  
@Bean  
open fun kotlinSupplier(): () -> String {  
    return { "Hello Kotlin" }  
}
```

[在这里](#)查看示例项目。

就是这样。该功能在当前快照中可用，并将成为 Spring Cloud Function 2.0.0.RELEASE 的一部分。这意味着增强和修改仍在进行中，因此您的反馈非常重要。

有关 Spring Cloud Function 的更多信息，请参阅以下内容：

[介绍 Spring Cloud Function 项目](#)

<https://www.nurkiewicz.com/2018/04/sneak-peek-at-spring-cloud-function.html>

# 参考资料

## 网站

- [spring-io] [spring.io](http://spring.io)
- [springdev-io] [springdev.io](http://springdev.io)
- [apache-org] [apache.org](http://apache.org)
- [sourceforge] [sourceforge.com](http://sourceforge.com)
- [github] [github.com](http://github.com)
- [csdn] [csdn.net](http://csdn.net)
- [infoq] [infoq.com](http://infoq.com)
- [oschina] [oschina.net](http://oschina.net)
- [openhub] [openhub.net](http://openhub.net)
- [baidu-index] [index.baidu.com](http://index.baidu.com)

## 网址

- [springone-platform-2018-look-back] <https://content.pivotal.io/pivotal-podcasts/a-look-back-at-springone-platform-2018-ep-80>
- [microsoft-to-acquire-github] <https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/>
- [coscon] <https://opensource.org/node/952>
- [source-han-serif] <https://source.typekit.com/source-han-serif/cn/>
- [china-open-source-report] <https://kaiyuanshe.github.io/2018-China-Open-Source-Report/>
- [gitlab-developer-survey-2018] <https://about.gitlab.com/developer-survey/2018/>
- [stackoverflow-survey-2018] <https://insights.stackoverflow.com/survey/2018/>
- [nodejs-user-survey-report] <https://nodejs.org/en/user-survey-report/>
- [jdk-11] <http://openjdk.java.net/projects/jdk/11/>
- [pvtl-nyse] <https://content.pivotal.io/announcements/pivotal-software-lists-on-nyse-as-pvtl>
- [ifenxi] <https://www.gelonghui.com/p/189174>

## 工具

- [asciidoc] <https://asciidoctor.org>
- [sublimetext] <https://www.sublimetext.com>
- [iterm2] <https://iterm2.com>
- [sketchapp] <https://sketchapp.com>
- [gradle] <https://gradle.org>
- [rake] <https://ruby.github.io/rake/>