慕课网
imooc.com

Broadview
www.broadview.com.cn

Spring Boot 编程思想（核心篇）

小马哥（mercyblitz）/ 著

昨夜西风凋碧树，独上高楼，望尽天涯路。

《蝶恋花》——晏殊

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Spring Boot 编程思想（核心篇）

小马哥（mercyblitz）/ 著

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

内 容 简 介

本书是《Spring Boot 编程思想》的核心篇，开篇总览 Spring Boot 核心特性，接着讨论自动装配（Auto-Configuration）与 SpringApplication。全书的讨论以 Spring Boot 为中心，议题发散至 Spring 技术栈、JSR 及 Java。希望透过全局的视角，帮助读者了解 Spring Boot 变迁的历程；经过多方的比较，帮助读者理解 Spring Boot 特性的原理；整合标准的规范，帮助读者掌握 Spring Boot 设计的哲学。

本书适合对 Spring Boot 感兴趣的读者阅读。

图书在版编目（CIP）数据

Spring Boot 编程思想：核心篇 / 小马哥著. —北京：电子工业出版社，2019.3
ISBN 978-7-121-36039-8

Ⅰ. ①S… Ⅱ. ①小… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2019）第 025869 号

责任编辑：陈晓猛
印　　刷：三河市华成印务有限公司
装　　订：三河市华成印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　印张：39.25　字数：879.2 千字
版　　次：2019 年 3 月第 1 版
印　　次：2019 年 4 月第 2 次印刷
定　　价：118.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

自序

非常感谢您阅读本书，在成长道路上，我们从此不再孤单。

大约在三年前，我有幸参与全集团微服务架构的演进及基础设施的构建，在此期间痛苦和受益并存。二〇一六年十二月，经朋友引荐，作为“SFDC 2016 杭州开发者大会”的嘉宾，进行了一场名为“微服务实践之路”的演讲，从此正式开始了我的微服务布道师之路。次年三月，segmentfault“讲堂”栏目上线，我再次受邀，作为 Java 讲师。同年六月二日，“Java 微服务实践”系列讲座正式直播，我主讲 Spring Boot 和 Spring Cloud。无独有偶，当月正好我工作满十周年，也萌生了著书的意向，计划写一本关于 Spring Boot 微服务开发实践的书籍，希望借此机会与诸君分享我的微服务实践经验。然而，随后的变故将此念头变为了现实。当月九日上午，正值当差，父亲传来一通电话，告知外婆于八点左右过世，听此噩耗，悲从中来，不可断绝。即刻带着身怀六甲的妻子，启程回湘。

外婆一直陪伴着我的成长，直到我远赴杭州求职，才分隔两地。现如今祖孙二人天各一方，生死茫茫，无处话凄凉，子欲养而亲不待的痛楚莫过于此。我曾向上天祷告，愿她能安享西方极乐。若非外婆的离世，我绝对不会有坚定意志和足够勇气来完成此书，书籍的内容也不会有颠覆性的变化，讨论的议题从过去的“Spring Boot 微服务开发实践”逐渐转变为“Spring Boot 编程思想”。希望竭尽所能，将技术积累、学习方法、实战经验，以及所思所想和盘托出。每当自己午夜梦回，脑海中浮现外婆的容貌时，总会潸然泪下，所有的思想动摇和行为慵懒立即烟消云散。外婆是虔诚的佛教徒，平日乐善好施。从小耳濡目染的我也尽一点绵薄之力，将书籍五成以上的稿费作为公益基金，支持贫困地区的青少年教育，并且不定期地公开账目信息，供广大读者监督。这或许杯水车薪，但仍希望他们能够感到一丝温暖。

祸兮福所倚，福兮祸所伏，生死轮回，自然之理。外婆去世后的两个月，我的儿子降临人间。作为一名新晋的父亲，自然会以更高的标准来要求自己，对书籍的质量同样趋于严苛，将早期已完成的部分“付之一炬”，推倒重来，内容篇幅剧增。作为我儿的表率，著书只是“立言”的开始，捐赠作为“立德”的发端，而“立行”则需身体力行，持之以恒。或许“著作等身”是一种不错的选择，然而现代科技的进步，尤其是文字载体的革新，要做到这一点，难度实在

不小。不过，“为者常成，行者常至”，实现从“小马哥”到“马三立”先生的华丽转身并非遥不可及。

已故南京大学历史系教授高华先生曾引述凯斯·詹京斯的观点，“历史乃论述过去，但绝不等于过去”。既然是论述，那么或多或少会存在偏差，不但受限于论述者的知识、能力及记忆等主观因素，而且取决于当时的时空环境。为了遵照原著，在功能特性的介绍上，本书将引述官方文档的英文原文，并做出适当的解释。由于文档的编写者或许不是代码的实现者，即使是实现者本人，也难免会站在自己的立场和高度，抑或章节安排及文字组织等诸多因素影响阅读和理解。因此，针对官方文档语焉不详的部分，本书将补充说明；对其错误的结论，将加以修正。由于本人能力和水平的局限，不敢妄言理解“格物致知”的奥义，难免有主观臆断和谬论之处，且仅一家之言，供诸君参考，切莫将此奉为圭臬，书云亦云，不假思索。老子有言，“上士闻道，勤而行之”，希望读者能学以致用，若能在实践中激发出创新的灵感，善莫大焉。

最后，借此机会，由衷地感谢我的太太，没有她背后默默地付出，我不会有如此多的精力投入，更无法专注写作。同时，向陈晓猛编辑致敬，他是一位谦谦君子，极富耐心，在书籍编写的过程中，给予我不少的帮助和鼓励。再次向各位朋友送上我诚挚的歉意，由于个人的原因，使得书籍出版时间一再跳票。

小马哥

公元二〇一九年一月于杭州

前言

本书全名为《Spring Boot 编程思想（核心篇）》，以 Spring Boot 2.0 为讨论的主线，讨论的范围将涵盖 Spring Boot 1.x 的所有版本，以及所关联的 Spring Framework 版本，致力于：

- 场景分析——掌握技术选型；
- 系统学习——拒绝浅尝辄止；
- 重视规范——了解发展趋势；
- 源码解读——理解设计思想；
- 实战演练——巩固学习成果。

内容总览

由于本书的内容跨度广，所以分“核心篇”“运维篇”和“Web 篇”三册分别讨论 Spring Boot 的功能特性。“核心篇”开篇总览 Spring Boot 核心特性，逐一讨论 Spring Boot 官网所罗列之六大特性，然而其中两点并非 Spring Boot 专属，故点到为止，而将讨论聚焦在其五大特性，分别为自动装配（Auto-Configuration）、SpringApplication、外部化配置、Spring Boot Actuator 和嵌入式 Web 容器。其中，前两者是“核心篇”讨论的议题，后两者则是 Spring Boot 官方定义的 Production-Ready 特性，均偏向 Spring Boot 应用运维，因此纳入“运维篇”的讨论范畴。至于嵌入式 Web 容器，将结合传统 Java EE Servlet、Spring Web MVC 和 Spring 5 WebFlux 的有关内容放至“Web 篇”探讨，具体章节安排如下。

- 核心篇
  - 总览 Spring Boot
  - 走向自动装配
  - 理解 SpringApplication

- 运维篇
  - 超越外部化配置
  - 简化 Spring 应用运维体系
- Web 篇
  - “渐行渐远”的 Servlet
  - 从 Servlet 到 Web MVC
  - 从 Reactive 到 WebFlux
  - 嵌入式 Web 容器

目前，“核心篇”和“运维篇”已编写完毕，“Web 篇”正在同步更新，其目录安排可能发生变更，请读者以最终发行版本为准。

在内容结构上，本书采用“总—分—总”的方式，首先总体介绍讨论范围，随后深入展开细节的讨论，最后予以总结。同时，为了避免先入为主的影响，本书将会针对官方文档的描述内容提出疑问或假设，大胆地猜测其可能实现的方式，再结合实现源码加以验证，随后通过示例代码巩固理解。在写作手法上，本书效仿中国历史书籍的传统编著手法，将纪传体和编年体予以综合。从功能特性来看，它属于纪传体，如自动装配、SpringApplication 和外部化配置等。如此表述方式更容易系统地介绍 Spring Boot 和 Spring Framework 的核心特性。从特性的发展历程来观察，它则属于编年体，如 Spring Framework 注解驱动编程模型从 1.x 到 5.0 的发展与 Spring Boot 自动装配之间的关联，以及 Spring Boot 1.0 到 1.4 的外部化配置源是怎样利用 Spring Environment 抽象逐步完善的。更重要的是，在论述方式上，增加了论点、论证和论据，从而做到知其然也知其所以然。在对特性的讨论中，会穿插一些补充说明。在特性讨论的结尾处，将总结所论议题。

所谓“兼听则明，偏听则暗”，本书的讨论范围并不会局限在 Spring Boot 或 Spring Framework，会将 Spring Cloud 甚至 Spring Cloud Data Flow 纳入参考，探讨 Spring Boot 在两者中的运用。站在更宏观的角度，在整个 Java EE 的生态中，Spring 技术栈并非独此一家，也不完全是“开山之作”，不少相关的特性可在 JSR 规范和其他 Java EE 实现中找到原型。换言之，Spring 技术栈可被认为是一种非常成功的“重复发明轮子”，不仅适配了 JSR 实现，而且“借鉴”了他山之石，逐步实现了自身的生态系统。

总而言之，全书的讨论将以 Spring Boot 为中心，议题发散至 Spring 技术栈、JSR 及 Java。希望读者透过全局的视角，了解变迁的历程；通过多方的比较，理解特性的原理；整合标准的规范，掌握设计的哲学。当您纵览全书之后，或许会明白为什么说“Spring Boot 易学难精”。因为它的核心是 Spring Framework，而对后者的理解程度又取决于对 JSR 规范及 Java 的熟悉度。

版本范围

为了系统性地讨论 Spring Boot 的发展脉络，本书会将 Spring Boot 2.0 与 1.x 的版本加以对比，探索从 1.0 到 2.0 版本的重要变化，便于读者后续架构、整合及迁移等工作。由于编写时间恰逢 Spring Boot 2.0.2.RELEASE 版本发布，为统一源码分析，将讨论的 Spring Boot 2.0 版本固定为 2.0.2.RELEASE，相同时间点的 Spring Boot 1.5 的版本则是 1.5.10.RELEASE。同时由于更低的版本已停止维护，所以可选择最后发行的小版本作为参考，故所有涉及的 Spring Boot 版本如下所示。

- Spring Boot 2.0：2.0.2.RELEASE；
- Spring Boot 1.5：1.5.10.RELEASE；
- Spring Boot 1.4：1.4.7.RELEASE；
- Spring Boot 1.3：1.3.8.RELEASE；
- Spring Boot 1.2：1.2.8.RELEASE；
- Spring Boot 1.1：1.1.9.RELEASE；
- Spring Boot 1.0：1.0.2.RELEASE。

由于 Spring Boot 2.0 最低依赖的 Java 版本为 8，而 Spring Boot 1.x 最低兼容 Java 1.6，因此 Java 1.8 是 Spring Boot 各个版本兼容的交集，也是实例工程 `thinking-in-spring-boot-samples` 的运行时环境。除此之外，讨论将更多地关注 Spring Boot 与其核心依赖 Spring Framework 之间的版本映射关系，如下表所示。

Spring Boot 版本	Spring Framework 版本	JDK 版本
2.0.2.RELEASE	5.0.6.RELEASE	1.8+
1.5.10.RELEASE	4.3.14.RELEASE	1.6+
1.4.7.RELEASE	4.3.9.RELEASE	1.6+
1.3.8.RELEASE	4.2.8.RELEASE	1.6+
1.2.8.RELEASE	4.1.9.RELEASE	1.6+
1.1.9.RELEASE	4.0.8.RELEASE	1.6+
1.0.2.RELEASE	4.0.3.RELEASE	1.6+

不难看出，Spring Boot 2.0 对应的 Spring Framework 版本是 5.0，而 Spring Boot 1.x 则依赖 Spring Framework 4.x。之所以要具体到 Spring Framework 的某个版本号，除了避免版本差异所导致源码分析失准的情况，更多的是由于 Spring 技术栈特殊的版本号管理。按照传统 Java 版本的约定，第一位数字表示主版本号，控制大版本更新，第二位代表次版本号，可小范围地引入新的特性和相关 API，而第三位则用于问题修正或安全补丁等。Spring 技术栈不时地利用第三位版本号引入新的

API，比如 Spring Framework 3.0.1 引入的 API `BeanDefinitionRegistryPostProcessor`，Spring Boot 1.3.2 引入的 API ExitCodeEvent。同时，在框架 API 的兼容性方面，从 Spring Framework 到 Spring Cloud 逐渐降低，Spring Boot 处于比上不足比下有余的状况。因此，本书在深入讨论的过程中反复地强调 API 兼容的重要性，也希望读者在自研的过程中多加关注。

为了理解 Spring Boot 特性发展的过程，将 Spring Framework 版本讨论的范围设定为从 1.x 到 5.0。换言之，本书将涵盖几乎所有的 Spring Framework 和 Spring Boot 版本，包括两者涉及的 JSR（Java Specification Requests），如 Servlet、Bean Validation 和 JAX-RS 等规范。

> 更多的 JSR 资讯，请参考官方网页 https://jcp.org/en/jsr/overview，或者访问小马哥 JSR 收藏页面 https://github.com/mercyblitz/jsr，下载归档的 JSR PDF 文件。

相关约定

本书在议题的讨论中，将在文档引用、示例代码、日志输出、源码版本、源码省略等方面做出约定。

文档引用约定

在文档引用方面，Spring Boot 官方文档的默认版本为 `2.0.2.RELEASE`，地址为：https://docs.spring.io/spring-boot/docs/2.0.2.RELEASE/reference/htmlsingle/。

Spring Framework 官方文档则选择 `5.0.6.RELEASE`，地址为：https//docs.spring.io/spring/docs/5.0.6.RELEASE/spring-framework-reference/。

为了遵照原文，讨论的过程中将援引原文，如 Spring Boot 官方文档在“11.5 Creating an Executable Jar”章节中介绍此等构建方式，即构建可执行 JAR，又称之为“fat jars”。

> We finish our example by creating a completely self-contained executable jar file that we could run in production. Executable jars (sometimes called “fat jars”) are archives containing your compiled classes along with all of the jar dependencies that your code needs to run.

此处的 Spring Boot 官方文档版本为 `2.0.2.RELEASE`，否则在讨论的内容中会特别说明其他版本信息。

示例代码约定

为了减少代码冗余和内容篇幅，通常会省略 Java 示例代码的 package 和 import 部分，如下所示。

```
@SpringBootApplication
public class FirstAppByGuiApplication {

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

    @Bean
    public RouterFunction<ServerResponse> helloWorld() {
        return route(GET("/hello-world"),
                request -> ok().body(Mono.just("Hello,World"), String.class)
        );
    }
}
```

如果该示例被重构或调整多次，则其不变的代码将被省略，仅关注变更或核心代码：

```
@SpringBootApplication
public class FirstAppByGuiApplication {
    ...

    /**
     * {@link ApplicationRunner#run(ApplicationArguments)}方法在
     * Spring Boot 应用启动后回调
     *
     * @param context WebServerApplicationContext
     * @return ApplicationRunner Bean
     */
    @Bean
    public ApplicationRunner runner(WebServerApplicationContext context) {
        return args -> {
            System.out.println("当前 WebServer 实现类为："
                    + context.getWebServer().getClass().getName());
```

```
        };
    }
}
```

不难发现，为了方便理解，在示例代码中会添加必要的注释加以说明。同时，如果在调整的过程中出现颠覆性变化，那么通常将注释不需要的功能，方便后续回顾：

```
//@Configuration
//@ComponentScan
@EnableAutoConfiguration
//@SpringBootApplication(scanBasePackages = "thinking.in.spring.boot.config")
public class FirstAppByGuiApplication {

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

//    /**
//     * {@link ApplicationRunner#run(ApplicationArguments)} 方法在
//     * Spring Boot 应用启动后回调
//     *
//     * @param context WebServerApplicationContext
//     * @return ApplicationRunner Bean
//     */
//    @Bean
//    public ApplicationRunner runner(WebServerApplicationContext context) {
//        return args -> {
//            System.out.println("当前 WebServer 实现类为: "
//                    + context.getWebServer().getClass().getName());
//        };
//    }
}
```

同样的原则适用于 XML 文件或其他配置文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
    <modelVersion>4.0.0</modelVersion>

    <groupId>thinking-in-spring-boot</groupId>
    <artifactId>first-app-by-gui</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    ...
        <!--&lt;!&ndash; Use Jetty instead &ndash;&gt;-->
        <!--<dependency>-->
            <!--<groupId>org.springframework.boot</groupId>-->
            <!--<artifactId>spring-boot-starter-jetty</artifactId>-->
        <!--</dependency>-->

        <!--&lt;!&ndash; Use Undertow instead &ndash;&gt;-->
        <!--<dependency>-->
            <!--<groupId>org.springframework.boot</groupId>-->
            <!--<artifactId>spring-boot-starter-undertow</artifactId>-->
        <!--</dependency>-->

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </dependency>
    ...
</project>
```

日志输出约定

为了精简示例运行时的日志输出，书中的内容并不一定完全与实际情况相同，主要的差异在于移除了重复及相关时间等非重要信息，例如：

```
$ mvn spring-boot:run
(...部分内容被省略...)
[           main] o.s.w.r.f.s.s.RouterFunctionMapping      : Mapped (GET &&
/hello-world) ->
thinkinginspringboot.firstappbygui.FirstAppByGuiApplication$$Lambda$276/708609190@7
af17431
(...部分内容被省略...)
```

```
当前 WebServer 实现类为：org.springframework.boot.web.embedded.tomcat.TomcatWebServer
[           main] t.f.FirstAppByGuiApplication            : Started
FirstAppByGuiApplication in 2.119 seconds (JVM running for 5.071)
```

当内容中出现“(...部分内容被省略...)”时，说明其中省略了数行的日志内容，并且几乎所有的日志输出到标准输出（System.out）。

源码版本约定

在源码分析过程中，为了理清不同版本中的实现细节和变化，通常在目标代码下方带有 Spring Boot 版本信息，以及 Maven 依赖的 GAV 坐标信息（GAV = groupId、artifactId 和 version），如下所示。

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@Configuration
@EnableAutoConfiguration
@ComponentScan
public @interface SpringBootApplication {

    /**
     * Exclude specific auto-configuration classes such that they will never be applied.
     * @return the classes to exclude
     */
    Class<?>[] exclude() default {};

}
```

> 以上实现源码源于 Spring Boot 1.2.8.RELEASE。
> Maven GAV 坐标为：org.springframework.boot:spring-boot-autoconfigure:1.2.8.RELEASE。

如果以上版本信息没有出现，则 Spring Boot 默认采用 2.0.2.RELEASE，Spring Framework 选择 5.0.6.RELEASE，JDK 源码的版本是 1.8.0_172。

源码省略约定

在源码分析的过程中，考虑到实现代码可能相对繁杂，为观其大意，便于记忆，会注释或移除部分无关痛痒的内容，例如：

```
public class AutoConfigurationImportSelector
        implements DeferredImportSelector, BeanClassLoaderAware, ResourceLoaderAware,
        BeanFactoryAware, EnvironmentAware, Ordered {
    ...
    protected List<String> getCandidateConfigurations(AnnotationMetadata metadata,
            AnnotationAttributes attributes) {
        List<String> configurations = SpringFactoriesLoader.loadFactoryNames(
                getSpringFactoriesLoaderFactoryClass(), getBeanClassLoader());
        ...
        return configurations;
    }
    ...
    protected Class<?> getSpringFactoriesLoaderFactoryClass() {
        return EnableAutoConfiguration.class;
    }
    ...
}
```

当然，其省略的部分并非一无是处，而是根据小马哥的个人经验来筛选的，必然受个人主观想法的影响，建议读者结合对应的版本源码，整体把握，逐步形成选读的意识。

表达约定

本书的讨论内容可能对相同事务出现不同的表述方式。

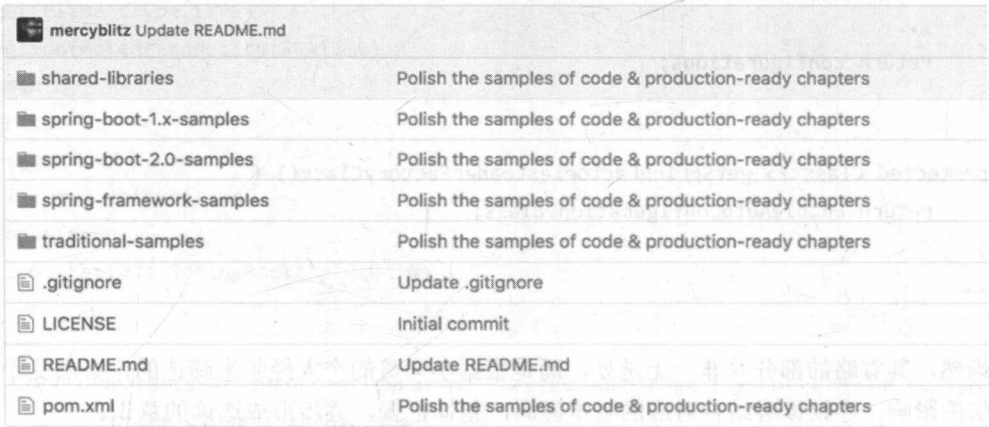
- 注解：Annotation；
- 配置 Class：@Configuration 类、@Configuration Class、Configuration Class；
- 包：package；
- 类路径：Class Path、class-path、类路径。

示例工程

本书所有的示例代码均存放在 https://github.com/mercyblitz/thinking-in-spring-boot-samples，该工程为标准的 Maven 多模块工程，运行时要求为 Java 1.8+和 Maven 3.2.5+。其协议为 Apache License Version 2.0，不必担心商业用途所带来的风险。由于本系列图书尚未完全定稿，工程结构未来可能存在微调。因此，当前内容无法确保百分之百匹配，请读者定期关注 README.md 文件，确保内容的更新。

工程结构

示例工程 `thinking-in-spring-boot-samples` 的结构如下图所示。



该工程包含五个子模块和四个文件，分别如下。

子模块及说明如下表所示。

子 模 块	说　明
shared-libraries	共享类库，为其他工程提供基础 API 或依赖
spring-boot-1.x-samples	Spring Boot 1.x 示例工程，包含六个子模块，主要用于参考和对比 Spring Boot 1.x 各版本的实现差异，并且提供章节示例代码实现
spring-boot-2.0-samples	Spring Boot 2.0 示例工程，也是主示例工程，以 2.0.2.RELEASE 作为基础版本
spring-framework-samples	Spring Framework 示例工程，作为 Spring Boot 底层实现框架，版本范围从 2.0 到 5.0
traditional-samples	传统 Java EE 示例工程，用于理解 Java EE 与 Spring Boot 的关联和差异

文件及其说明如下表所示。

文　件	说　明
.gitignore	Git 版本控制文件
LICENSE	工程许可文件
README.md	工程说明文件
pom.xml	示例工程 Maven pom.xml 文件

其中，又以 spring-boot-2.0-samples、spring-boot-1.x-samples 和 spring-framework-samples 为本示例工程最核心的子模块，对此将详细说明。

子模块 spring-boot-2.0-samples

spring-boot-2.0-samples 作为《Spring Boot 编程思想（核心篇）》的主示例工程，基于 Spring Boot `2.0.2.RELEASE` 实现，由若干子模块组成，这些模块与章节所讨论的议题紧密关联：

```
├── spring-boot-2.0-samples
│   ├── auto-configuration-sample
│   ├── externalized-configuration-sample
│   ├── first-app-by-gui
│   ├── first-spring-boot-application
│   ├── formatter-spring-boot-starter
│   ├── pom.xml
│   ├── production-ready-sample
│   ├── spring-application-sample
│   ├── spring-boot-2.0-samples.iml
│   └── traditional-web-sample
```

按照本书的安排，模块与章节所对应的关系如下表所示。

子 模 块	说　明	篇　章
first-spring-boot-application	基于 Maven 插件构建的第一个 Spring Boot 应用	核心篇——总览 Spring Boot
first-app-by-gui	基于图形化界面 https://start.spring.io/构建的第一个 Spring Boot 应用	核心篇——总览 Spring Boot
auto-configuration-sample	Spring Boot 自动装配示例	核心篇——走向自动装配
formatter-spring-boot-starter	Spring Boot 自动装配 Starter 示例	核心篇——走向自动装配
spring-application-sample	Spring Boot SpringApplication 示例	核心篇——理解 SpringApplication

续表

子 模 块	说　明	篇　章
externalized-configuration-sample	Spring Boot 外部化配置示例	运维篇——超越外部化配置
production-ready-sample	Spring Boot Production-Ready	运维篇——简化 Spring 应用运维体系
traditional-web-sample	Spring Boot 应用部署到传统 Servlet 容器示例	Web 篇——“渐行渐远”的 Servlet

除此之外，相关示例代码部分也可能放置在其他子模块，如“走向自动装配”章节中，大量的实例代码在子模块 spring-framework-samples 之中。

子模块 spring-boot-1.x-samples

前文提到，该子模块包含六个子模块，它们对应了所有的 Spring Boot 1.x 实现版本：

```
├── spring-boot-1.x-samples
│   ├── pom.xml
│   ├── spring-boot-1.0.x-project
│   ├── spring-boot-1.1.x-project
│   ├── spring-boot-1.2.x-project
│   ├── spring-boot-1.3.x-project
│   ├── spring-boot-1.4.x-project
│   ├── spring-boot-1.5.x-project
```

截至当前编写时间，恰逢 Spring Boot 1.5 的发行版本为 `1.5.10.RELEASE`，而 1.5 之前的版本已停止维护，可选择其最后发行的版本作为参考，故子模块、Spring Boot 1.x 版本及对应 Spring Framework 的关系如下表所示。

子 模 块	Spring Boot 1.x 版本	Spring Framework 版本
spring-boot-1.0.x-project	1.0.2.RELEASE	4.0.3.RELEASE
spring-boot-1.1.x-project	1.1.9.RELEASE	4.0.8.RELEASE
spring-boot-1.2.x-project	1.2.8.RELEASE	4.1.9.RELEASE
spring-boot-1.3.x-project	1.3.8.RELEASE	4.2.8.RELEASE
spring-boot-1.4.x-project	1.4.7.RELEASE	4.3.9.RELEASE
spring-boot-1.5.x-project	1.5.10.RELEASE	4.3.14.RELEASE

值得注意的是，子模块 spring-boot-1.x-samples 并非主示例工程，各章节讨论的特性示例并非面面俱到。相反，读者应重点关注子模块 spring-boot-2.0-samples。