[Sping boot 参考指南 2](#_Toc7072)

[第一部分 Spring Boot 文档 2](#_Toc1180)

[1. 关于文档 2](#_Toc21309)

[2. 获取帮助 2](#_Toc4814)

[3. 第一步 3](#_Toc14641)

[4. 使用Spring Boot 3](#_Toc8307)

[5. 了解Spring Boot 特性 3](#_Toc32345)

[6. 转移到生产环境 3](#_Toc211)

[7. 高级主题 4](#_Toc24402)

[第二部分、 入门指南 4](#_Toc8896)

[8. Spring Boot介绍 4](#_Toc18854)

[9. 系统要求 4](#_Toc6037)

[9.1 Servlet 容器 5](#_Toc25428)

[10. 安装Spring Boot 5](#_Toc10014)

# Sping boot 参考指南

**作者**

Phillip Webb, Dave Syer, Josh Long, Stéphane Nicoll, Rob Winch, Andy Wilkinson, Marcel Overdijk, Christian Dupuis, Sébastien Deleuze,Michael Simons, Vedran Pavić, Jay Bryant, Madhura Bhave

2.0.4.RELEASE

Copyright © 2012-2018

本文件的副本可供您自己使用并分发给他人，前提是您不对此类副本收取任何费用并进一步提供每份副本均包含此版权声明，无论是以印刷版还是电子版分发。

## Spring Boot 文档

这部分介绍spring Boot文档的简要概述。它作为其他部分文档的映射。

### 关于文档

Spring Boot文档可用作

* HTML
* PDF
* EPUB

最新副本可在以下网站获得 [docs.spring.io/spring-boot/docs/current/reference]( docs.spring.io/spring-boot/docs/current/reference)

本文档的副本可供您自己使用并分发给他人，前提是您不对此类副本收取任何费用

并进一步规定，每份副本均包含本版权声明，无论是以印刷版还是电子版分发。

### 获取帮助

<https://docs.spring.io/spring-boot/docs/2.0.4.RELEASE/reference/htmlsingle/>

如果你对Spring boot有问题，我们愿意帮忙

* 尝试How-to documents。它提供最常见问题的解决方法
* 学习Spring基础。Spring boot建立在许多其他的Spring项目上。查看spring.io网站上的丰富的参考文档。如果你不看Spring基础文档，看第一条
* 提问。我们监控stackoverflow.com网站上用spring-boot标签标记的问题。
* 在github.com/spring-projects/spring-boot/issues网站上提交关于Spring Boot的bug

所有的Spring Boot内容是开源的，包括文档，如果你发现文档有什么问题并想要改进它，请参与进来。

### 第一步

如果你开始使用Spring Boot或 Spring，从下面几项开始。

* 从头开始：Overview | Requirements | Installation
* 教程：Part1 | Part2
* 运行你的范例：Part1 | Part2

### 使用Spring Boot

准备开始实操Spring Boot？我们已经打包了以下内容

* 构建系统: Maven | Gradle | Ant | Starters
* 最佳做法：Code Structure | @Configuration | @EnableAutoConfiguration | Beans and Dependency Injection
* 运行代码：IDE | Packaged | Maven | Gradle
* 打包应用程序： Production jars
* Spring Boot CLI：使用CLI

### 了解Spring Boot 特性

需要更多Spring Boot的核心细节？我们为你准备了以下内容。

* 核心功能: SpringApplication | External Configuration | Profiles | Logging
* Web 应用: MVC | Embedded Containers
* 数据处理: SQL | NO-SQL
* 消息处理: Overview | JMS
* 测试: Overview | Boot Applications | Utils
* 扩展: Auto-configuration | @Conditions

### 转移到生产环境

当您准备好将Spring Boot 应用程序放到生产环境时，我们有一些您可能会喜欢的技巧！

* 管理终端: Overview | Customization
* 连接选项： HTTP | JMX
* 监控: Metrics | Auditing | Tracing | Process

### 高级主题

最后，我们还有一些面向高级用户的主题：

* 发布Spring Boot 应用： Cloud Deployment | OS Service
* 构建工具插件： Maven | Gradle
* 附录： Application Properties | Auto-configuration classes | Executable Jars

## 入门指南

如果您开始使用Spring Boot或一般的“Spring”，请首先阅读本节。 它回答了基本的“什么？”，“如何？”和“为什么？”的问题。 它包括Spring Boot简介以及安装说明。 然后，我们将引导您构建您的第一个Spring启动应用程序并讨论一些核心原则。

### Spring Boot介绍

Spring Boot可以基于Spring简单地创建独立的、生产级别的可运行项目。我们对Spring平台和第三方库有自己的看法，这样你可以尽量少的瞎忙活。大多数的Spring Boot应用需要很少的Spring配置。

你可以用Spring Boot创建java应用程序，通过java-jar或一些传统的打包方式发布。我们也提供运行命令行的“spring scripts”。

我们的主要目标：

* 为所有的spring开发者提供一个更快、更广泛可接受的入门体验
* 开箱即用，当需求开始出现偏离时快速发生转变
* 提供大量项目中常见的一系列非功能特征（例如嵌入式服务器，安全性，指标，运行状况检查，外部化配置）。
* 绝对没有代码生成，也不需要XML配置。

### 系统要求

Spring Boot 2.0.4.RELEASE需要Java 8 或 9 ，Spring Framework 5.0.8.RELEASE 或以上版本。 为Maven（3.2+）、Gradle 4提供了显式构建支持。

#### Servlet 容器

Spring Boot 支持如下嵌入式servlet容器:

Name Servlet Version

Tomcat 8.5 3.1

Jetty 9.4 3.1

Undertow 1.4 3.1

你也可以将Spring Boot应用发布到任何兼容Servlet 3.1的容器中。

### 安装Spring Boot

Spring boot 可以与“经典”java开发工具一起使用或作为命令行工具安装。你需要java SDK1.8或更高版本。在你开始之前，你需要检查当前的安装，使用一下命令：

$ java -version

如果你是java开发新手或者你想要体验Spring Boot，你也许想要首先尝试Spring Boot CLI，

如果想正式使用Spring Boot，请阅读“经典(classic)”安装说明。

#### 针对Java开发程序员安装说明

使用Spring Boot和任何版本标本java库相同，只需要在类路径下包含适当的spring-boot-\*.jar文件。Spring Boot不需要任何的特殊集成工具，你可以使用任何的IDE或者文本编辑器。Spring Boot应用也没有什么特殊的地方，因此你可以像任何其他的java程序一样运行和调试Spring

Boot应用。

虽然你可以直接复制Spring Boot的jar包，但是我们通常建议你使用依赖关系管理的构建工具（比如Maven或Gradle）。

##### Maven 安装

Spring Boot 兼容Apache Maven 3.2及以上版本，如果你还没有安装Maven，你可以按照maven.apache.org上的说明来安装。

在很多操作系统上，Maven可以通过软件包管理器进行安装。如果您是OSX Homebrew用户，请尝试使用命令：brew install maven。 Ubuntu用户可以运行命令：sudo apt-get install maven。Windows使用Chocolatey 可以运行choco安装maven。

Spring Boot 依赖 org.springframework.boot groupId。通常，您的Maven POM文件将从 spring-boot-starter-parent 项目继承，并声明一个或多个“启动器(启动器)”的依赖关系。Spring Boot还提供了一个可选的Maven插件来创建可执行的jar包。

下面列出一个典型的pom.xml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>com.example</groupId>  <artifactId>myproject</artifactId>  <version>0.0.1-SNAPSHOT</version>  <!-- Inherit defaults from Spring Boot -->  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.4.RELEASE</version>  </parent>  <!-- Add typical dependencies for a web application -->  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  </dependencies>  <!-- Package as an executable jar -->  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

Spring-boot-starter-parent是使用Spring Boot的一个很好的方式，但是它可能并不适合所有的情况。有时候你可能需要从不同的父Pom继承，或者你也许不喜欢我们的默认设置。在这些情况下，看13.2.2章，“使用不带父Pom的Spring Boot”作为使用导入作用域(import scope)的替代解决方案。

##### 10.1.2 Gradle 安装

Spring Boot 兼容 Gradle 2（2.9或更高版本）和Gradle 3。如果您尚未安装Gradle，您可以按照 [http://www.gradle.org/](http://www.gradle.org/" \t "http://blog.geekidentity.com/spring/spring_boot_translation/_blank) 上的说明进行操作。

可以使用org.springframework.boot 组(group)声明Spring Boot 的依赖项。 通常，您的项目将声明一个或多个“启动器(Starters)”的依赖。Spring Boot提供了一个有用的Gradle插件，可用于简化依赖关系声明和创建可执行 jar包。

Gradle Wrapper

当您需要构建项目时，Gradle Wrapper提供了一种“获取(obtaining)”Gradle的更好的方式。 它是一个小脚本和库，它与代码一起引导构建过程。 有关详细信息，请参阅 [https://docs.gradle.org/2.14.1/userguide/gradle\_wrapper.html](https://docs.gradle.org/2.14.1/userguide/gradle_wrapper.html" \t "http://blog.geekidentity.com/spring/spring_boot_translation/_blank) 。

典型的 build.gradle 文件：

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 | plugins {  id 'org.springframework.boot' version '1.5.2.RELEASE'  id 'java' }   jar {  baseName = 'myproject'  version = '0.0.1-SNAPSHOT' }  repositories {  jcenter() }  dependencies {  compile("org.springframework.boot:spring-boot-starter-web")  testCompile("org.springframework.boot:spring-boot-starter-test") } |

#### 安装Spring Boot CLI

Spring Boot CLI（命令行接口）是一个命令行工具，可以使用它进行Spring的快速原型（quickly prototype）。它允许你运行Groovy脚本，这意味着会有你熟悉的类似java的语法，没有太多的样板代码（boilerplate code）。

你不必要通过CLI来使用Spring Boot，但它绝对是spring应用最快的落地方式。

##### 10.2.1 手动安装

你可以从Spring软件版本库下载Spring CLI发行版：

spring-boot-cli-2.0.4.RELEASE-bin.zip

spring-boot-cli-2.0.4.RELEASE-bin.tar.gz

还提供前沿快照发行版

下载后，按照解压后文件INSTALL.txt的介绍进行操作。总而言之：在.zip文件的bin/目录中有一个spring脚本（Windows的spring.bat），或者你可以使用java -jar（脚本可以帮助您确保类路径设置正确）。

##### 10.2.2 使用 SDKMAN!安装

SDKMAN! (软件开发套件管理器) 可以管理各种二进制SDK的多个版本, 包括 Groovy 和

the Spring Boot CLI. 从 sdkman.io 获取SDKMAN!并安装Spring Boot ，使用如下命令:

|  |
| --- |
| $ sdk install springboot  $ spring --version  Spring Boot v2.0.4.RELEASE |

如果你正在开发CLI的功能，并希望轻松访问刚创建的版本，使用下面的命令：

|  |
| --- |
| $sdk install springboot dev /path/to/spring-boot/spring-boot-cli/target/spring-boot-cli-2.0.4.RELEASE-bin/spring-2.0.4.RE  $ sdk default springboot dev  $ spring --version  Spring CLI v2.0.4.RELEASE |

前面的命令安装了一个名为dev的spring本地实例。它指向你构建位置的target，所以每次你重新构建（rebulid）Spring Boot，spring将是最新的。

运行一下命令你可以看到：

|  |
| --- |
| $ sdk ls springboot  ===============================================================================  Available Springboot Versions  ===============================================================================  > + dev  \* 1.5.2.RELEASE  ===============================================================================  + - local version  \* - installed  > - currently in use  =============================================================================== |

##### 10.2.3 OSX Homebrew 安装

如果您在Mac上使用 Homebrew，安装Spring Boot CLI 只需要下面命令：

|  |
| --- |
| $ brew tap pivotal/tap  $ brew install springboot |

Homebrew会将Spring 安装到 /usr/local/bin。

如果您没有看到公式(formula)，您的安装可能会过期。 只需执行brew更新，然后重试。

##### 10.2.4 MacPorts安装

如果您在Mac上使用 MacPorts，安装Spring Boot CLI 只需要下面命令：

$ sudo port install spring-boot-cli

##### 10.2.5 命令行提示

Spring Boot CLI为BASH和zsh shell提供命令提示的功能。 您可以在任何shell中引用脚本（也称为spring），或将其放在您的个人或系统范围的bash完成初始化中。 在Debian系统上，系统范围的脚本位于 /shell-completion/bash 中，当新的shell启动时，该目录中的所有脚本将被执行。 手动运行脚本，例如 如果您使用SDKMAN安装了！

$ . ~/.sdkman/candidates/springboot/current/shell-completion/bash/spring

$ spring <HIT TAB HERE>

grab help jar run test version

如果使用Homebrew或MacPorts安装Spring Boot CLI，则命令行补全脚本将自动注册到您的shell。

##### 10.2.6 Windows Scoop 安装

如果你在Windows平台使用Scoop，你可以用如下命令安装Spring Boot CLI：

> scoop bucket add extras

> scoop install springboot

Scoop 把spring安装到 to ~/scoop/apps/springboot/current/bin .

如果你没有看到app的表现，你安装的scoop也许过期了。在这种情况下，更新scoop再试。

##### 10.2.7 快速启动Spring CLI示例

这是一个非常简单的Web应用程序，可用于测试您的安装是否正确。创建一个名为app.groovy的文件：

@RestController

class ThisWillActuallyRun {

@RequestMapping("/")

String home() {

"Hello World!"

}

}

然后从shell运行它：

$ spring run app.groovy

因为下载依赖的库，首次运行应用程序需要一些时间，后续运行将会更快。

在浏览器中打开 http://localhost:8080 ，您应该会看到以下输出：

Hello World!

#### 从早期版本的Spring Boot升级

如果你要从更早的版本升级Spring Boot，请检查项目wiki上托管的“迁移指南”，还要检查“发行说明”以及每个版本“新的和值得注意的”特性。

当升级到一个新的特性发布版本，有些属性可能被重命名或者移除。Spring Boot提供一个方法去分析你的应用程序的环境和启动的打印诊断，但是也可能在运行时临时迁移属性。为了启用这个特性，给你的项目增加如下依赖：

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-properties-migrator</artifactId>  <scope>runtime</scope>  </dependency> |

在环境中添加的属性（例如使用@PropertySource时）将不会被考虑在内。

完成迁移后，请确保从项目的依赖项中删除此模块。

为了升级已经存在的CLI安装，使用是让的包管理命令（比如，brew upgrade）或者，如果你手动安装了CLI，按照标准说明操作，更新PATH环境变量移除任何旧引用。

### 开发你的第一个Spring Boot 应用程序

这章描述了怎样开发一个简单的“Hello World!” web应用并凸显了一些Spring Boot的核心特点。我们使用Maven去构建项目，因为大部分IDEs支持它。

Spring.io网站包含了许多Spring Boot入门指南。如果你需要解决一个具体问题，先去这里查找。

你可以通过转到spring.io并从依赖关系搜索器中选择“web”启动器来快捷执行以下步骤。

这样做会生成一个新的项目结构，以便您可以立即开始编码。 有关更多详细信息，请查看Spring Initializr文档。

在我们开始之前，打开一个终端并且运行一下命令确保安装了有效的java和maven版本。

|  |
| --- |
| $ java -version  java version "1.8.0\_102"  Java(TM) SE Runtime Environment (build 1.8.0\_102-b14)  Java HotSpot(TM) 64-Bit Server VM (build 25.102-b14, mixed mode)  $ mvn -v  Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T16:41:47+00:00)  Maven home: /usr/local/Cellar/maven/3.3.9/libexec  Java version: 1.8.0\_102, vendor: Oracle Corporation |

#### 11.1 创建 POM

我们需要创建Maven的pom.xml文件。Pom.xml是用来构建项目的配方。打开你最爱的文本编辑器添加一下内容：

|  |
| --- |
| <?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>com.example</groupId>  <artifactId>myproject</artifactId>  <version>0.0.1-SNAPSHOT</version>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.4.RELEASE</version>  </parent>  <!-- Additional lines to be added here... -->  </project> |

上面的清单能为你提供有效的构建。运行mvn package测试（现在，你可以忽略“jar将是空的 - 没有内容标记为包含!”的警告）

此时，您可以将项目导入IDE（大多数现代Java IDE包括对Maven的内置支持）。 为简单起见我们继续为此示例使用纯文本编辑器。

#### 11.2 添加Classpath依赖

Spring Boot提供了许多“Starters”，可以将jar添加到类路径中。我们的示例应用程序已经在Pom的父节点使用了 spring-boot-starter-parent。 spring-boot-starter-parent是一个特殊的启动器，提供有用的maven默认值。它还提供依赖管理 dependency-management 标签，以便您可以省略子模块依赖关系的版本标签。

其他“启动器(Starters)”只是提供您在开发特定类型的应用程序时可能需要的依赖关系。

由于我们正在开发Web应用程序，所以我们将添加一个spring-boot-starter-web依赖关系，但在此之前，我们来看看我们目前的依赖。

$ mvn dependency:tree  
[INFO] com.example:myproject:jar:0.0.1-SNAPSHOT

**The mvn dependency:tree** 打印项目依赖关系的树形表示

**spring-boot-starter-parent** 本身不在依赖关系中。

编辑 pom.xml 文件，并在parent 下添加**spring-boot-starter-web** 依赖

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

如果您再次运行 mvn dependency:tree ，您将看到现在有许多附加依赖关系，包括Tomcat Web服务器和Spring Boot本身。

#### 11.3 编写代码

要完成我们的应用程序，我们需要创建一个的Java文件。 默认情况下，Maven将从src/main/java编译源代码，因此您需要创建该文件夹结构，然后添加一个名为src/main/java/Example.java的文件：

import org.springframework.boot.\*;  
import org.springframework.boot.autoconfigure.\*;  
import org.springframework.stereotype.\*;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
@EnableAutoConfiguration  
public class Example {  
  
 @RequestMapping("/")  
 String home() {  
 return "Hello World!";  
 }  
  
 public static void main(String[] args) throws Exception {  
 SpringApplication.run(Example.class, args);  
 }  
  
}

虽然这里没有太多的代码，但是有一些重要的部分。

##### 11.3.1 @RestController和@RequestMapping 注解

我们的Example类的第一个注解是@RestController。 这被称为 stereotype annotation。它为人们阅读代码提供了一些提示，对于Spring来说，这个类具有特定的作用。在这里，我们的类是一个web @Controller，所以Spring在处理传入的Web请求时会考虑这个类。

@RequestMapping注解提供“路由”信息。 告诉Spring，任何具有路径“/”的HTTP请求都应映射到home方法。 @RestController注解告诉Spring将生成的字符串直接返回给调用者。

@RestController和@RequestMapping注解是Spring MVC 的注解（它们不是Spring Boot特有的）。 有关更多详细信息，请参阅Spring参考文档中的MVC部分。

##### 11.3.2 @EnableAutoConfiguration注解

第二个类级别的注释是@EnableAutoConfiguration。 这个注解告诉 Spring Boot 根据您添加的jar依赖关系来“猜(guess)”你将如何配置Spring。由于spring-boot-starter-web添加了Tomcat和Spring MVC，自动配置将假定您正在开发Web应用程序并相应地配置Spring。

启动器和自动配置

自动配置旨在与“启动器”配合使用，但两个概念并不直接相关。 您可以自由选择启动器之外的jar依赖项，Spring Boot仍然会自动配置您的应用程序。

##### 11.3.3 “main”方法

我们的应用程序的最后一部分是main()方法。 这只是一个遵循Java惯例的应用程序入口点的标准方法。 我们的main()方法通过调用run()委托(delegates)给Spring Boot的SpringApplication类。 SpringApplication将引导我们的应用程序，启动Spring，然后启动自动配置的Tomcat Web服务器。 我们需要将Example.class作为一个参数传递给run方法来告诉SpringApplication，它是主要的Spring组件。 还传递了args数组以传递命令行参数。