
Table of Contents

| | |
|--------------------|-----|
| 内容简介 | 1.1 |
| 目录 | 1.2 |
| 序言 | 1.3 |
| 第一章 微服务和SpringBoot | 1.4 |
| 第二章 环境搭建 | 1.5 |
| 第三章 从HellWorld开始 | 1.6 |
| 第四章 第一个Web页面 | 1.7 |
| 第五章 使用velocity模板引擎 | 1.8 |

简明SpringBoot教程:SpringBoot一路向西

针对新手的SpringBoot入门级简明实例教程. (亮剑)

关于作者:

刘鼎亮，花名帛书，江西客家人，热爱技术，热爱理财，热爱简单的生活。IT工程师。

陈光剑，花名一剑，江苏东海人，号行走江湖一剑客，字之剑。程序员，诗人，作家。

一直致力于基于SpringBoot的Web平台开发。

目录

微服务和**SpringBoot**

环境搭建

从**HelloWorld**开始

第一个**Web**页面

使用**Velocity**模板引擎

前端请求后端服务

使用**JPA**操作数据库

部署

SpringBoot与**Scala**

序言

目前，SpringBoot技术在业内特别受追捧，但国内关于Spring Boot的书籍少之又少。即使能找到一两本，那也是一些技术大牛从他们的高角度去看待和讲解Spring Boot，一般初学者不容易理解和消化。

本书借以此，从零基础的角度，一步一步介绍Spring Boot微服务框架的使用。书中从HelloWorld开始，由浅入深，一步一步展开讲解。希望对初学者有帮助。也希望本书可以成为初学者的《简明Spring Boot教程》。

本书介绍的是Web平台开发黄金搭配：IntelliJ IDEA + Maven + Spring Boot + Velocity + Bootstrap + jQuery.

第一章 微服务和SpringBoot

微服务架构

“微服务架构”这个术语最近几年横空出世，它以若干组可独立部署的服务的方式进行软件应用系统的设计。关于微服务架构的讨论最早是一篇由Martin Fowler在2014写的著名文章开始的。尽管这种架构风格在业界尚无精确的定义，但Martin Fowler还是给出了以下特征描述：它是以开发一组小型服务的方式来开发一个独立的应用系统的。其中每个小型服务都运行在自己的进程中，并经常采用HTTP资源API这样轻量的机制来相互通信。这些服务围绕业务功能进行构建，并能通过全自动的部署机制来进行独立部署。这些微服务可以使用不同的语言来编写，并且可以使用不同的数据存储技术。对这些微服务我们仅做最低限度的集中管理。

说白了，微服务架构相比传统架构的一个显著特点就是去中心化，将之前的服务再细分，并配套相应的自动化基础设施。

SpringBoot

根据ThoughtWorks2016年最新的一次技术雷达显示，SpringBoot微服务框架越来越得到业界的重视和应用。之前的Spring框架在Java语言的带动下已经在企业开发中广泛使用，如果Spring生态系统正像技术雷达中的显示，走向微服务架构。那SpringBoot就是当下最好的选择。

SpringBoot就是为微服务而诞生的，而且它提供了一个强大的一键式Spring的集成开发环境，能够单独进行一个Spring应用的开发，其中：

- (1) 集中式配置（application.properties）+注解，大大简化了开发流程
- (2) 内嵌的Tomcat和Jetty容器，可直接打成jar包启动，无需提供Java war包以及繁琐的Web配置
- (3) 提供了Spring各个插件的基于Maven的pom模板配置，开箱即用，便利无比。
- (4) 可以在任何你想自动化配置的地方，实现可能
- (5) 提供更多的企业级开发特性，如何系统监控，健康诊断，权限控制
- (6) 无冗余代码生成和XML强制配置
- (7) 提供支持强大的Restful风格的编码，非常简洁

当然Spring Boot提供的功能，远远比上面的强大。Spring boot集成了servlet容器，当我们在pom文件中增加spring-boot-starter-web的maven依赖时，不做任何web相关的配置便能提供web服务，这还得归于Spring boot自动配置的功能（因为加了EnableAutoConfiguration的注

解)，帮我们创建了一堆默认的配置，以前在web.xml中配置，现在都可以通过spring bean的方式进行配置，由spring来进行生命周期的管理，大多数情况下，我们需要重载这些配置（例如修改服务的启动端口，contextpath,filter,listener,servlet,session超时时间等）

第二章 环境搭建

JDK

推荐使用最新版本

Maven

到官网下载：<https://maven.apache.org/>

IDE

根据个人喜爱，如IntelliJ, Eclipse都是不错的选择。

第三章 从HelloWorld开始

创建Maven工程

打开IntelliJ或Eclipse，新建一个Maven工程（Project），其中GroupId和ArtifactId可以随便填写，如Hello

如果不清楚如何创建Maven工程，请大家自动百度。

创建好后(笔者创建的名称为HelloWorld)，如果使用的是IntelliJ IDE，它会弹出Maven projects need to be imported提示，选择“Enable Auto-Import”。

工程目录结构为：

HelloWorld --src --main --java --resources --test（里面内容可以忽略） --pom.xml

配置pom.xml文件

打开工程名下的pom.xml，原内容如下：<?xml version="1.0" encoding="UTF-8"?>

4.0.0 hello hello 1.0-SNAPSHOT

1、添加parent标签，以下指明SpringBoot版本为1.3.6

org.springframework.boot spring-boot-starter-parent 1.3.6.RELEASE

2、添加starter-web依赖，因为我们是开发Web平台，所以必须依赖spring-boot-starter-web，它内置了tomcat容器

org.springframework.boot spring-boot-starter-web


添加后完整的pom.xml文件内容为：

<?xml version="1.0" encoding="UTF-8"?>

4.0.0 hello hello 1.0-SNAPSHOT org.springframework.boot spring-boot-starter-parent

1.3.5.RELEASE org.springframework.boot spring-boot-starter-web

创建入口类

创建包名 展开src->main->java，右击java，新建一个package，名称为lightsword。右击lightsword，新建一个java类，名称为lightsword.java，如下图所示：

lightsword.java内容如下：（增加一个main方法，调用SpringApplication类的run方法，引用了一个SpringBootApplication注解）

package lightsword;


```
import org.springframework.boot.SpringApplication; import
org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication public class lightsword { public static void main(String[] args){
SpringApplication.run(lightsword.class, args); } }
```

运行lightsword.java后，出现以下信息表示运行成功：

```
. _ ^ / ' 0 \ \ \ \ ( ( ) _ | ' | 7 | ' V ` | \ \ \ \ v ) | D | I | I | I | I | ( I ) ) ) ) ' [ ] . # I I I I , | / / / /  
=====||=====|_ /= // _ / .... : Started lightsword in 3.953 seconds (JVM  
running for 5.172)
```

为了打印**Hello World**，我们在**main**方法加入一行代码 `public static void main(String[] args){
SpringApplication.run(lightsword.class, args); System.out.println("Hello world"); }`

再次运行，结果多了一行Hello world: : Started lightsword in 3.908 seconds (JVM running for 4.95) Hello world

可能遇到的问题：端口冲突。如果你电脑上的8080端口号被其它程序（如jenkins）占用了，则运行lightsword会报以下错误：`java.net.BindException: Address already in use Failed to start component [Connector[HTTP/1.1-8080]] ...`

解决方法：在src->main->resources目录下新建一个文件，名称为application.properties（这是SpringBoot统一的配置文件）加了以下一行内容：（取个电脑上可用的端口号，如下面的9527，看过星爷电影的都懂的） server.port = 9527

再次运行即可。

讲解

分析我们的入口类lightsword.java。

1. **@SpringBootApplication**注解，它其实帮我们做了很多事情，包括各种配置，扫描等。它相当于以下三个注解的作用：**@Configuration @EnableAutoConfiguration @ComponentScan** 为了简化和方便，我们必须在入口类加入以上注解，当然最方便的就使用**@SpringBootApplication**一个注解。
2. 调用**SpringApplication**类的**run**启动应用程序。

后面章节我们都使用**9527**端口作为讲解。

第四章 第一个**Web**页面