



# Spring Boot



**tutorialspoint**

SIMPLY EASY LEARNING

[www.tutorialspoint.com](http://www.tutorialspoint.com)



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

## About the Tutorial

---

Spring Boot is an open source Java-based framework used to create a Micro Service. It is developed by Pivotal Team. It is easy to create a stand-alone and production ready spring applications using Spring Boot. Spring Boot contains a comprehensive infrastructure support for developing a microservice and enables you to develop enterprise-ready applications that you can “**just run**”.

## Audience

---

This tutorial is designed for Java developers to understand and develop production-ready spring applications with minimum configurations. It explores major features of Spring Boot such as Starters, Auto-configuration, Beans, Actuator and more.

By the end of this tutorial, you will gain an intermediate level of expertise in Spring Boot.

## Prerequisites

---

This tutorial is written for readers who have a prior experience of Java, Spring, Maven, and Gradle. You can easily understand the concepts of Spring Boot if you have knowledge on these concepts. It would be an additional advantage if you have an idea about writing a RESTful Web Service. If you are a beginner, we suggest you to go through tutorials related to these concepts before you start with Spring Boot.

## Copyright and Disclaimer

---

© Copyright 2018 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at [contact@tutorialspoint.com](mailto:contact@tutorialspoint.com)

## Table of Contents

---

About the Tutorial .....	i
Audience.....	i
Prerequisites.....	i
Copyright and Disclaimer .....	i
Table of Contents.....	ii
 1. SPRING BOOT – INTRODUCTION .....	 1
What is Micro Service?.....	1
What is Spring Boot?.....	1
Why Spring Boot? .....	2
How does it work? .....	2
Spring Boot Starters .....	2
Auto Configuration .....	3
Spring Boot Application .....	4
Component Scan.....	4
 2. SPRING BOOT – QUICK START .....	 6
Prerequisites.....	6
Spring Boot CLI.....	6
 3. SPRING BOOT – BOOTSTRAPPING .....	 8
Spring Initializer .....	8
Maven.....	9
Gradle.....	10
Class Path Dependencies .....	11
Main Method.....	12
Write a Rest Endpoint .....	12
Create an Executable JAR.....	13

Run Hello World with Java .....	14
4. SPRING BOOT – TOMCAT DEPLOYMENT .....	16
Spring Boot Servlet Initializer .....	16
Setting Main Class .....	17
Update packaging JAR into WAR .....	17
Packaging your Application .....	19
Deploy into Tomcat .....	20
5. SPRING BOOT – BUILD SYSTEMS .....	25
Dependency Management .....	25
Maven Dependency .....	25
Gradle Dependency .....	26
6. SPRING BOOT – CODE STRUCTURE .....	27
Default package .....	27
Typical Layout .....	27
6. SPRING BOOT – CODE STRUCTURE .....	27
7. SPRING BOOT – SPRING BEANS AND DEPENDENCY INJECTION .....	28
8. SPRING BOOT – RUNNERS .....	29
Application Runner .....	29
Command Line Runner .....	30
9. SPRING BOOT – APPLICATION PROPERTIES .....	31
Command Line Properties .....	31
Properties File .....	31
YAML File .....	31
Externalized Properties .....	32
Use of @Value Annotation .....	32

Spring Boot Active Profile .....	33
10. SPRING BOOT – LOGGING .....	37
Log Format.....	37
Console Log Output.....	37
File Log Output .....	37
Log Levels .....	38
Configure Logback.....	38
11. SPRING BOOT – BUILDING RESTFUL WEB SERVICES .....	41
Rest Controller.....	43
Request Mapping.....	43
Request Body.....	44
Path Variable .....	44
Request Parameter .....	44
GET API .....	44
POST API .....	46
PUT API.....	47
DELETE API.....	48
12. SPRING BOOT – EXCEPTION HANDLING .....	54
Controller Advice .....	54
ExceptionHandler.....	54
13. SPRING BOOT – INTERCEPTOR .....	62
14. SPRING BOOT – SERVLET FILTER.....	71
15. SPRING BOOT – TOMCAT PORT NUMBER .....	77
Custom Port.....	77
Random Port.....	77

16. SPRING BOOT – REST TEMPLATE.....	78
GET .....	79
POST .....	80
PUT .....	81
DELETE .....	82
17. SPRING BOOT – FILE HANDLING.....	89
File Upload.....	89
File Download .....	90
18. SPRING BOOT – SERVICE COMPONENTS .....	96
19. SPRING BOOT – THYMELEAF .....	106
Thymeleaf Templates.....	106
Web Application .....	106
20. SPRING BOOT – CONSUMING RESTFUL WEB SERVICES.....	113
Angular JS .....	122
21. SPRING BOOT – CORS SUPPORT .....	124
Enable CORS in Controller Method .....	124
Global CORS Configuration .....	124
22. SPRING BOOT – INTERNATIONALIZATION .....	126
Dependencies .....	126
LocaleResolver .....	126
LocaleChangeInterceptor .....	127
Messages Sources .....	127
HTML file .....	128
23. SPRING BOOT – SCHEDULING .....	134
Java Cron Expression.....	134

Fixed Rate .....	135
Fixed Delay .....	136
24. SPRING BOOT – ENABLING HTTPS .....	138
Self-Signed Certificate .....	138
Configure HTTPS .....	139
25. SPRING BOOT – EUREKA SERVER .....	140
Building a Eureka Server .....	140
26. SPRING BOOT – SERVICE REGISTRATION WITH EUREKA .....	146
27. SPRING BOOT – ZUUL PROXY SERVER AND ROUTING .....	153
Creating Zuul Server Application .....	153
28. SPRING BOOT – SPRING CLOUD CONFIGURATION SERVER .....	160
Creating Spring Cloud Configuration Server .....	160
29. SPRING BOOT – SPRING CLOUD CONFIGURATION CLIENT .....	166
Working with Spring Cloud Configuration Server .....	166
30. SPRING BOOT – ACTUATOR .....	169
Enabling Spring Boot Actuator .....	169
31. SPRING BOOT – ADMIN SERVER .....	171
32. SPRING BOOT – ADMIN CLIENT .....	176
33. SPRING BOOT – ENABLING SWAGGER2 .....	179
34. SPRING BOOT – CREATING DOCKER IMAGE .....	186
Create Dockerfile .....	186
Maven .....	186
Gradle .....	190
35. SPRING BOOT – TRACING MICRO SERVICE LOGS .....	194

Spring Cloud Sleuth .....	194
Zipkin Server .....	199
36. SPRING BOOT – FLYWAY DATABASE.....	206
Configuring Flyway Database .....	206
37. SPRING BOOT – SENDING EMAIL.....	213
38. SPRING BOOT – HYSTRIX .....	219
39. SPRING BOOT – WEB SOCKET.....	226
40. SPRING BOOT – BATCH SERVICE.....	235
41. SPRING BOOT – SPRING FOR APACHE KAFKA .....	244
Producing Messages .....	244
Consuming a Message.....	245
42. SPRING BOOT – TWILIO.....	252
Sending SMS .....	252
Voice Calls.....	257
43. SPRING BOOT – UNIT TEST CASES .....	262
Mockito .....	262
44. SPRING BOOT – REST CONTROLLER UNIT TEST .....	269
Writing a Unit Test for REST Controller .....	269
45. SPRING BOOT – DATABASE HANDLING .....	276
Connect to H2 database .....	276
Connect MySQL.....	277
Connect Redis .....	279
JdbcTemplate.....	280
Multiple DataSource .....	280



46. SPRING BOOT – SECURING WEB APPLICATIONS.....	284
Securing a Web application.....	284
47. SPRING BOOT SECURITY – OAUTH2 WITH JWT .....	294
Authorization Server .....	294
Resource Server .....	294
OAuth2 .....	294
JWT Token .....	294
48. SPRING BOOT – GOOGLE CLOUD PLATFORM .....	311
Google Cloud SQL.....	314
49. SPRING BOOT – GOOGLE OAUTH2 SIGN-IN.....	316

# 1. Spring Boot – Introduction

Spring Boot is an open source Java-based framework used to create a micro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. This chapter will give you an introduction to Spring Boot and familiarizes you with its basic concepts.

## What is Micro Service?

---

Micro Service is an architecture that allows the developers to develop and deploy services independently. Each service running has its own process and this achieves the lightweight model to support business applications.

### Advantages

Micro services offers the following advantages to its developers:

- Easy deployment
- Simple scalability
- Compatible with Containers
- Minimum configuration
- Lesser production time

## What is Spring Boot?

---

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can **just run**. You can get started with minimum configurations without the need for an entire Spring configuration setup.

### Advantages

Spring Boot offers the following advantages to its developers:

- Easy to understand and develop spring applications
- Increases productivity
- Reduces the development time

### Goals

Spring Boot is designed with the following goals:

- To avoid complex XML configuration in Spring
- To develop a production ready Spring applications in an easier way
- To reduce the development time and run the application independently
- Offer an easier way of getting started with the application

## Why Spring Boot?

---

You can choose Spring Boot because of the features and benefits it offers as given here:

- It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
- It provides a powerful batch processing and manages REST endpoints.
- In Spring Boot, everything is auto configured; no manual configurations are needed.
- It offers annotation-based spring application
- Eases dependency management
- It includes Embedded Servlet Container

## How does it work?

---

Spring Boot automatically configures your application based on the dependencies you have added to the project by using **@EnableAutoConfiguration** annotation. For example, if MySQL database is on your classpath, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.

The entry point of the spring boot application is the class contains **@SpringBootApplication** annotation and the main method.

Spring Boot automatically scans all the components included in the project by using **@ComponentScan** annotation.

## Spring Boot Starters

---

Handling dependency management is a difficult task for big projects. Spring Boot resolves this problem by providing a set of dependencies for developers convenience.

For example, if you want to use Spring and JPA for database access, it is sufficient if you include **spring-boot-starter-data-jpa** dependency in your project.

Note that all Spring Boot starters follow the same naming pattern **spring-boot-starter-\***, where \* indicates that it is a type of the application.

## Examples

Look at the following Spring Boot starters explained below for a better understanding:

**Spring Boot Starter Actuator dependency** is used to monitor and manage your application. Its code is shown below:

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

**Spring Boot Starter Security dependency** is used for Spring Security. Its code is shown below:

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

**Spring Boot Starter web dependency** is used to write a Rest Endpoints. Its code is shown below:

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

**Spring Boot Starter Thyme Leaf dependency** is used to create a web application. Its code is shown below:

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

**Spring Boot Starter Test dependency** is used for writing Test cases. Its code is shown below:

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

## Auto Configuration

Spring Boot Auto Configuration automatically configures your Spring application based on the JAR dependencies you added in the project. For example, if MySQL database is on your class path, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.

For this purpose, you need to add **@EnableAutoConfiguration** annotation or **@SpringBootApplication** annotation to your main class file. Then, your Spring Boot application will be automatically configured.

Observe the following code for a better understanding:

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

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

## Spring Boot Application

The entry point of the Spring Boot Application is the class contains **@SpringBootApplication** annotation. This class should have the main method to run the Spring Boot application. **@SpringBootApplication** annotation includes Auto-Configuration, Component Scan, and Spring Boot Configuration.

If you added **@SpringBootApplication** annotation to the class, you do not need to add the **@EnableAutoConfiguration**, **@ComponentScan** and **@SpringBootConfiguration** annotation. The **@SpringBootApplication** annotation includes all other annotations.

Observe the following code for a better understanding:

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

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

## Component Scan

Spring Boot application scans all the beans and package declarations when the application initializes. You need to add the **@ComponentScan** annotation for your class file to scan your components added in your project.

Observe the following code for a better understanding:

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

```
import org.springframework.context.annotation.ComponentScan;

@ComponentScan
public class DemoApplication {

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

## 2. Spring Boot – Quick Start

This chapter will teach you how to create a Spring Boot application using Maven and Gradle.

### Prerequisites

---

Your system need to have the following minimum requirements to create a Spring Boot application:

- Java 7
- Maven 3.2
- Gradle 2.5

### Spring Boot CLI

---

The Spring Boot CLI is a command line tool and it allows us to run the Groovy scripts. This is the easiest way to create a Spring Boot application by using the Spring Boot Command Line Interface. You can create, run and test the application in command prompt itself.

This section explains you the steps involved in manual installation of Spring Boot CLI . For further help, you can use the following link: <https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#getting-started-installing-spring-boot>

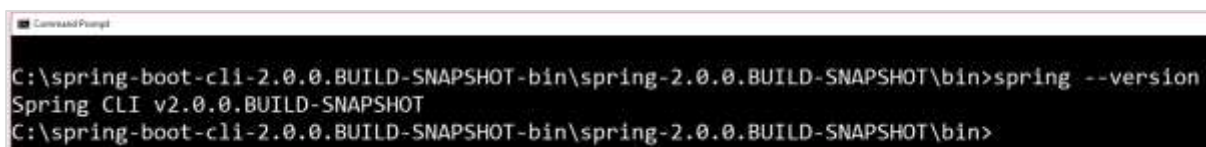
You can also download the Spring CLI distribution from the Spring Software repository at: <https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#getting-started-manual-cli-installation>

For manual installation, you need to use the following two folders:

- **spring-boot-cli-2.0.0.BUILD-SNAPSHOT-bin.zip**
- **spring-boot-cli-2.0.0.BUILD-SNAPSHOT-bin.tar.gz**

After the download, unpack the archive file and follow the steps given in the install.txt file. Not that it does not require any environment setup.

In Windows, go to the Spring Boot CLI **bin** directory in the command prompt and run the command **spring --version** to make sure spring CLI is installed correctly. After executing the command, you can see the spring CLI version as shown below:



```
Command Prompt
C:\spring-boot-cli-2.0.0.BUILD-SNAPSHOT-bin\spring-2.0.0.BUILD-SNAPSHOT\bin>spring --version
Spring CLI v2.0.0.BUILD-SNAPSHOT
C:\spring-boot-cli-2.0.0.BUILD-SNAPSHOT-bin\spring-2.0.0.BUILD-SNAPSHOT\bin>
```