**Introducing Spring Boot**

**Introducing Spring Boot**

Spring Boot makes it easy to create stand-alone, production-grade Spring-based Applications that you can run. We take an opinionated view of the Spring platform and third-party libraries, so that you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

You can use Spring Boot to create Java applications that can be started by using java -jar or more traditional war deployments. We also provide a command line tool that runs “spring scripts”.

Our primary goals are:

* Provide a radically faster and widely accessible getting-started experience for all Spring development.
* Be opinionated out of the box but get out of the way quickly as requirements start to diverge from the defaults.
* Provide a range of non-functional features that are common to large classes of projects (such as embedded servers, security, metrics, health checks, and externalized configuration).
* Absolutely no code generation and no requirement for XML configuration.

**Introducing Spring Boot**

Spring Boot makes it easy to create stand-alone, production-grade Spring-based Applications that you can run. We take an opinionated view of the Spring platform and third-party libraries, so that you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

You can use Spring Boot to create Java applications that can be started by using java -jar or more traditional war deployments. We also provide a command line tool that runs “spring scripts”.

Our primary goals are:

* Provide a radically faster and widely accessible getting-started experience for all Spring development.
* Be opinionated out of the box but get out of the way quickly as requirements start to diverge from the defaults.
* Provide a range of non-functional features that are common to large classes of projects (such as embedded servers, security, metrics, health checks, and externalized configuration).
* Absolutely no code generation and no requirement for XML configuration.

## 10. Installing Spring Boot

Spring Boot can be used with “classic” Java development tools or installed as a command line tool. Either way, you need [Java SDK v1.8](https://www.java.com/) or higher. Before you begin, you should check your current Java installation by using the following command:

$ java -version

If you are new to Java development or if you want to experiment with Spring Boot, you might want to try the [Spring Boot CLI](https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html#getting-started-installing-the-cli) (Command Line Interface) first. Otherwise, read on for “classic” installation instructions.

## 10.1 Installation Instructions for the Java Developer

You can use Spring Boot in the same way as any standard Java library. To do so, include the appropriate spring-boot-\*.jar files on your classpath. Spring Boot does not require any special tools integration, so you can use any IDE or text editor. Also, there is nothing special about a Spring Boot application, so you can run and debug a Spring Boot application as you would any other Java program.

Although you could copy Spring Boot jars, we generally recommend that you use a build tool that supports dependency management (such as Maven or Gradle).

### 10.1.1 Maven Installation

Spring Boot is compatible with Apache Maven 3.3 or above. If you do not already have Maven installed, you can follow the instructions at [maven.apache.org](https://maven.apache.org/).

|  |
| --- |
| [Tip] |
| On many operating systems, Maven can be installed with a package manager. If you use OSX Homebrew, try brew install maven. Ubuntu users can run sudo apt-get install maven. Windows users with [Chocolatey](https://chocolatey.org/) can run choco install maven from an elevated (administrator) prompt. |

Spring Boot dependencies use the org.springframework.boot groupId. Typically, your Maven POM file inherits from the spring-boot-starter-parent project and declares dependencies to one or more [“Starters”](https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-build-systems.html#using-boot-starter). Spring Boot also provides an optional [Maven plugin](https://docs.spring.io/spring-boot/docs/current/reference/html/build-tool-plugins-maven-plugin.html) to create executable jars.

The following listing shows a typical pom.xml file:

<?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>2.1.3.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>

|  |
| --- |
| [Tip] |
| The spring-boot-starter-parent is a great way to use Spring Boot, but it might not be suitable all of the time. Sometimes you may need to inherit from a different parent POM, or you might not like our default settings. In those cases, see [Section 13.2.2, “Using Spring Boot without the Parent POM”](https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-build-systems.html#using-boot-maven-without-a-parent) for an alternative solution that uses an import scope. |

### 10.1.2 Gradle Installation

Spring Boot is compatible with Gradle 4.4 and later. If you do not already have Gradle installed, you can follow the instructions at [gradle.org](https://gradle.org/).

Spring Boot dependencies can be declared by using the org.springframework.boot group. Typically, your project declares dependencies to one or more [“Starters”](https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-build-systems.html#using-boot-starter). Spring Boot provides a useful [Gradle plugin](https://docs.spring.io/spring-boot/docs/current/reference/html/build-tool-plugins-gradle-plugin.html" \o "72. Spring Boot Gradle Plugin) that can be used to simplify dependency declarations and to create executable jars.

**Gradle Wrapper**

The Gradle Wrapper provides a nice way of “obtaining” Gradle when you need to build a project. It is a small script and library that you commit alongside your code to bootstrap the build process. See [docs.gradle.org/4.2.1/userguide/gradle\_wrapper.html](https://docs.gradle.org/4.2.1/userguide/gradle_wrapper.html) for details.

More details on getting started with Spring Boot and Gradle can be found in the [Getting Started section](https://docs.spring.io/spring-boot/docs/2.1.3.RELEASE/gradle-plugin/reference/html/#getting-started) of the Gradle plugin’s reference guide.

## 10.2 Installing the Spring Boot CLI

The Spring Boot CLI (Command Line Interface) is a command line tool that you can use to quickly prototype with Spring. It lets you run [Groovy](http://groovy-lang.org/) scripts, which means that you have a familiar Java-like syntax without so much boilerplate code.

You do not need to use the CLI to work with Spring Boot, but it is definitely the quickest way to get a Spring application off the ground.

### 10.2.1 Manual Installation

You can download the Spring CLI distribution from the Spring software repository:

* [spring-boot-cli-2.1.3.RELEASE-bin.zip](https://repo.spring.io/release/org/springframework/boot/spring-boot-cli/2.1.3.RELEASE/spring-boot-cli-2.1.3.RELEASE-bin.zip)
* [spring-boot-cli-2.1.3.RELEASE-bin.tar.gz](https://repo.spring.io/release/org/springframework/boot/spring-boot-cli/2.1.3.RELEASE/spring-boot-cli-2.1.3.RELEASE-bin.tar.gz)

Cutting edge [snapshot distributions](https://repo.spring.io/snapshot/org/springframework/boot/spring-boot-cli/) are also available.

Once downloaded, follow the [INSTALL.txt](https://raw.github.com/spring-projects/spring-boot/v2.1.3.RELEASE/spring-boot-project/spring-boot-cli/src/main/content/INSTALL.txt) instructions from the unpacked archive. In summary, there is a spring script (spring.bat for Windows) in a bin/ directory in the .zip file. Alternatively, you can use java -jar with the .jar file (the script helps you to be sure that the classpath is set correctly).

### 10.2.2 Installation with SDKMAN!

SDKMAN! (The Software Development Kit Manager) can be used for managing multiple versions of various binary SDKs, including Groovy and the Spring Boot CLI. Get SDKMAN! from [sdkman.io](http://sdkman.io/) and install Spring Boot by using the following commands:

$ sdk install springboot

$ spring --version

Spring Boot v2.1.3.RELEASE

If you develop features for the CLI and want easy access to the version you built, use the following commands:

$ sdk install springboot dev /path/to/spring-boot/spring-boot-cli/target/spring-boot-cli-2.1.3.RELEASE-bin/spring-2.1.3.RELEASE/

$ sdk default springboot dev

$ spring --version

Spring CLI v2.1.3.RELEASE

The preceding instructions install a local instance of spring called the dev instance. It points at your target build location, so every time you rebuild Spring Boot,spring is up-to-date.

You can see it by running the following command:

$ sdk ls springboot

================================================================================

Available Springboot Versions

================================================================================

> + dev

\* 2.1.3.RELEASE

================================================================================

+ - local version

\* - installed

> - currently in use

================================================================================

### 10.2.3 OSX Homebrew Installation

If you are on a Mac and use [Homebrew](https://brew.sh/), you can install the Spring Boot CLI by using the following commands:

$ brew tap pivotal/tap

$ brew install springboot

Homebrew installs spring to /usr/local/bin.

|  |
| --- |
| [Note] |
| If you do not see the formula, your installation of brew might be out-of-date. In that case, run brew update and try again. |

### 10.2.4 MacPorts Installation

If you are on a Mac and use [MacPorts](https://www.macports.org/" \t "_top), you can install the Spring Boot CLI by using the following command:

$ sudo port install spring-boot-cli

### 10.2.5 Command-line Completion

The Spring Boot CLI includes scripts that provide command completion for the [BASH](https://en.wikipedia.org/wiki/Bash_%28Unix_shell%29) and [zsh](https://en.wikipedia.org/wiki/Z_shell" \t "_top) shells. You can source the script (also named spring) in any shell or put it in your personal or system-wide bash completion initialization. On a Debian system, the system-wide scripts are in /shell-completion/bash and all scripts in that directory are executed when a new shell starts. For example, to run the script manually if you have installed by using SDKMAN!, use the following commands:

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

$ spring <HIT TAB HERE>

grab help jar run test version

|  |
| --- |
| [Note] |
| If you install the Spring Boot CLI by using Homebrew or MacPorts, the command-line completion scripts are automatically registered with your shell. |

### 10.2.6 Windows Scoop Installation

If you are on a Windows and use [Scoop](http://scoop.sh/), you can install the Spring Boot CLI by using the following commands:

> scoop bucket add extras

> scoop install springboot

Scoop installs spring to ~/scoop/apps/springboot/current/bin.

|  |
| --- |
| [Note] |
| If you do not see the app manifest, your installation of scoop might be out-of-date. In that case, run scoop update and try again. |

### 10.2.7 Quick-start Spring CLI Example

You can use the following web application to test your installation. To start, create a file called app.groovy, as follows:

*@RestController*

**class** ThisWillActuallyRun {

*@RequestMapping("/")*

String home() {

"Hello World!"

}

}

Then run it from a shell, as follows:

$ spring run app.groovy

|  |
| --- |
| [Note] |
| The first run of your application is slow, as dependencies are downloaded. Subsequent runs are much quicker. |

Open [localhost:8080](http://localhost:8080/) in your favorite web browser. You should see the following output:

Hello World!

## 10.3 Upgrading from an Earlier Version of Spring Boot

If you are upgrading from an earlier release of Spring Boot, check the [“migration guide” on the project wiki](https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.0-Migration-Guide) that provides detailed upgrade instructions. Check also the[“release notes”](https://github.com/spring-projects/spring-boot/wiki) for a list of “new and noteworthy” features for each release.

When upgrading to a new feature release, some properties may have been renamed or removed. Spring Boot provides a way to analyze your application’s environment and print diagnostics at startup, but also temporarily migrate properties at runtime for you. To enable that feature, add the following dependency to your project:

<dependency>

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

<artifactId>spring-boot-properties-migrator</artifactId>

<scope>runtime</scope>

</dependency>

|  |
| --- |
| [Warning] |
| Properties that are added late to the environment, such as when using @PropertySource, will not be taken into account. |
| [Note] |
| Once you’re done with the migration, please make sure to remove this module from your project’s dependencies. | |

To upgrade an existing CLI installation, use the appropriate package manager command (for example, brew upgrade) or, if you manually installed the CLI, follow the[standard instructions](https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html#getting-started-manual-cli-installation), remembering to update your PATH environment variable to remove any older references.

## 11. Developing Your First Spring Boot Application

This section describes how to develop a simple “Hello World!” web application that highlights some of Spring Boot’s key features. We use Maven to build this project, since most IDEs support it.

|  |
| --- |
|  |
|  |

Before we begin, open a terminal and run the following commands to ensure that you have valid versions of Java and Maven installed:

$ 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 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T14:33:14-04:00)

Maven home: /usr/local/Cellar/maven/3.3.9/libexec

Java version: 1.8.0\_102, vendor: Oracle Corporation

|  |
| --- |
| [Note] |
| This sample needs to be created in its own folder. Subsequent instructions assume that you have created a suitable folder and that it is your current directory. |

## 11.1 Creating the POM

We need to start by creating a Maven pom.xml file. The pom.xml is the recipe that is used to build your project. Open your favorite text editor and add the following:

<?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>2.1.3.RELEASE</version>

</parent>

*<!-- Additional lines to be added here... -->*

</project>

The preceding listing should give you a working build. You can test it by running mvn package (for now, you can ignore the “jar will be empty - no content was marked for inclusion!” warning).

|  |
| --- |
| [Note] |
| At this point, you could import the project into an IDE (most modern Java IDEs include built-in support for Maven). For simplicity, we continue to use a plain text editor for this example. |

## 11.2 Adding Classpath Dependencies

Spring Boot provides a number of “Starters” that let you add jars to your classpath. Our sample application has already used spring-boot-starter-parent in the parent section of the POM. The spring-boot-starter-parent is a special starter that provides useful Maven defaults. It also provides a [dependency-management](https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-build-systems.html#using-boot-dependency-management)section so that you can omit version tags for “blessed” dependencies.

Other “Starters” provide dependencies that you are likely to need when developing a specific type of application. Since we are developing a web application, we add aspring-boot-starter-web dependency. Before that, we can look at what we currently have by running the following command:

$ mvn dependency:tree

[INFO] com.example:myproject:jar:0.0.1-SNAPSHOT

The mvn dependency:tree command prints a tree representation of your project dependencies. You can see that spring-boot-starter-parent provides no dependencies by itself. To add the necessary dependencies, edit your pom.xml and add the spring-boot-starter-web dependency immediately below the parentsection:

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

If you run mvn dependency:tree again, you see that there are now a number of additional dependencies, including the Tomcat web server and Spring Boot itself.

## 11.3 Writing the Code

To finish our application, we need to create a single Java file. By default, Maven compiles sources from src/main/java, so you need to create that folder structure and then add a file named src/main/java/Example.java to contain the following code:

**import** org.springframework.boot.\*;

**import** org.springframework.boot.autoconfigure.\*;

**import** org.springframework.web.bind.annotation.\*;

*@RestController*

*@EnableAutoConfiguration*

**public** **class** Example {

*@RequestMapping("/")*

String home() {

**return** "Hello World!";

}

**public** **static** **void** main(String[] args) {

SpringApplication.run(Example.**class**, args);

}

}

Although there is not much code here, quite a lot is going on. We step through the important parts in the next few sections.

### 11.3.1 The @RestController and @RequestMapping Annotations

The first annotation on our Example class is @RestController. This is known as a stereotype annotation. It provides hints for people reading the code and for Spring that the class plays a specific role. In this case, our class is a web @Controller, so Spring considers it when handling incoming web requests.

The @RequestMapping annotation provides “routing” information. It tells Spring that any HTTP request with the / path should be mapped to the home method. The@RestController annotation tells Spring to render the resulting string directly back to the caller.

|  |
| --- |
| [Tip] |
| The @RestController and @RequestMapping annotations are Spring MVC annotations. (They are not specific to Spring Boot.) See the [MVC section](https://docs.spring.io/spring/docs/5.1.5.RELEASE/spring-framework-reference/web.html#mvc) in the Spring Reference Documentation for more details. |

### 11.3.2 The @EnableAutoConfiguration Annotation

The second class-level annotation is @EnableAutoConfiguration. This annotation tells Spring Boot to “guess” how you want to configure Spring, based on the jar dependencies that you have added. Since spring-boot-starter-web added Tomcat and Spring MVC, the auto-configuration assumes that you are developing a web application and sets up Spring accordingly.

**Starters and Auto-configuration**

Auto-configuration is designed to work well with “Starters”, but the two concepts are not directly tied. You are free to pick and choose jar dependencies outside of the starters. Spring Boot still does its best to auto-configure your application.

### 11.3.3 The “main” Method

The final part of our application is the main method. This is just a standard method that follows the Java convention for an application entry point. Our main method delegates to Spring Boot’s SpringApplication class by calling run. SpringApplication bootstraps our application, starting Spring, which, in turn, starts the auto-configured Tomcat web server. We need to pass Example.class as an argument to the run method to tell SpringApplication which is the primary Spring component. The args array is also passed through to expose any command-line arguments.

## 11.4 Running the Example

At this point, your application should work. Since you used the spring-boot-starter-parent POM, you have a useful run goal that you can use to start the application. Type mvn spring-boot:run from the root project directory to start the application. You should see output similar to the following:

$ mvn spring-boot:run

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

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

:: Spring Boot :: (v2.1.3.RELEASE)

....... . . .

....... . . . (log output here)

....... . . .

........ Started Example in 2.222 seconds (JVM running for 6.514)

If you open a web browser to [localhost:8080](http://localhost:8080/), you should see the following output:

Hello World!

To gracefully exit the application, press ctrl-c.

## 11.5 Creating an Executable Jar

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.

**Executable jars and Java**

Java does not provide a standard way to load nested jar files (jar files that are themselves contained within a jar). This can be problematic if you are looking to distribute a self-contained application.

To solve this problem, many developers use “uber” jars. An uber jar packages all the classes from all the application’s dependencies into a single archive. The problem with this approach is that it becomes hard to see which libraries are in your application. It can also be problematic if the same filename is used (but with different content) in multiple jars.

Spring Boot takes a [different approach](https://docs.spring.io/spring-boot/docs/current/reference/html/executable-jar.html) and lets you actually nest jars directly.

To create an executable jar, we need to add the spring-boot-maven-plugin to our pom.xml. To do so, insert the following lines just below the dependenciessection:

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

|  |
| --- |
| [Note] |
| The spring-boot-starter-parent POM includes <executions> configuration to bind the repackage goal. If you do not use the parent POM, you need to declare this configuration yourself. See the [plugin documentation](https://docs.spring.io/spring-boot/docs/2.1.3.RELEASE/maven-plugin/usage.html) for details. |

Save your pom.xml and run mvn package from the command line, as follows:

$ mvn package

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building myproject 0.0.1-SNAPSHOT

[INFO] ------------------------------------------------------------------------

[INFO] .... ..

[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ myproject ---

[INFO] Building jar: /Users/developer/example/spring-boot-example/target/myproject-0.0.1-SNAPSHOT.jar

[INFO]

[INFO] --- spring-boot-maven-plugin:2.1.3.RELEASE:repackage (default) @ myproject ---

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

If you look in the target directory, you should see myproject-0.0.1-SNAPSHOT.jar. The file should be around 10 MB in size. If you want to peek inside, you can use jar tvf, as follows:

$ jar tvf target/myproject-0.0.1-SNAPSHOT.jar

You should also see a much smaller file named myproject-0.0.1-SNAPSHOT.jar.original in the target directory. This is the original jar file that Maven created before it was repackaged by Spring Boot.

To run that application, use the java -jar command, as follows:

$ java -jar target/myproject-0.0.1-SNAPSHOT.jar

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

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

:: Spring Boot :: (v2.1.3.RELEASE)

....... . . .

....... . . . (log output here)

....... . . .

........ Started Example in 2.536 seconds (JVM running for 2.864)

As before, to exit the application, press ctrl-c.