9. Installing Spring Boot

Spring Boot can be used with ``classic'' Java development tools or installed as a command line tool. Regardless, you will need Java SDK v1.6 or higher. You should check your current Java installation before you begin:

```
$ java -version
```

If you are new to Java development, or if you just want to experiment with Spring Boot you might want to try the Spring Boot CLI first, otherwise, read on for ``classic'' installation instructions.

Although Spring Boot is compatible with Java 1.6, if possible, you should consider using the latest version of Java.

9.1 Installation instructions for the Java developer

You can use Spring Boot in the same way as any standard Java library. Simply 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; and there is nothing special about a Spring Boot application, so you can run and debug as you would any other Java program.

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

9.1.1 Maven installation

Spring Boot is compatible with Apache Maven 3.0 or above. If you don't already have Maven installed you can follow the instructions at maven.apache.org.

On many operating systems Maven can be installed via a package manager. If you're an OSX Homebrew user try `brew install maven`. Ubuntu users can run `sudo apt-get install maven`.

Spring Boot dependencies use the `org.springframework.boot` `groupId`. Typically your Maven POM file will inherit from the `spring-boot-starter-parent` project and declare dependencies to one or more ``Starter POMs''. Spring Boot also provides an optional Maven plugin to create executable jars.

Here is 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>1.2.0.M2</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>

    <!-- Add Spring repositories -->
    <!-- (you don't need this if you are using a .RELEASE version) -->
    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <url>http://repo.spring.io/snapshot</url>
            <snapshots><enabled>true</enabled></snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <url>http://repo.spring.io/milestone</url>
        </repository>
    </repositories>
    <pluginRepositories>
        <pluginRepository>
            <id>spring-snapshots</id>
            <url>http://repo.spring.io/snapshot</url>
        </pluginRepository>
        <pluginRepository>
            <id>spring-milestones</id>
            <url>http://repo.spring.io/milestone</url>
        </pluginRepository>
    </pluginRepositories>
</project>
```

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 just not like our default settings. See Section 12.1.2, “Using Spring Boot without the parent POM” for an alternative solution that uses an `import` scope.

9.1.2 Gradle installation

Spring Boot is compatible with Gradle 1.6 or above. If you don’t already have Gradle installed you can follow the instructions at www.gradle.org/.

Spring Boot dependencies can be declared using the `org.springframework.boot` `group`. Typically your project will declare dependencies to one or more ``Starter POMs''. Spring Boot provides a useful 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’s a small script and library that you commit alongside your code to bootstrap the build process. See www.gradle.org/docs/current/userguide/gradle_wrapper.html for details.

Here is a typical `build.gradle` file:

```
buildscript {
    repositories {
```

```
        jcenter()
        maven { url "http://repo.spring.io/snapshot" }
        maven { url "http://repo.spring.io/milestone" }
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:1.2.0.M2")
    }
}

apply plugin: 'java'
apply plugin: 'spring-boot'

jar {
    baseName = 'myproject'
    version =  '0.0.1-SNAPSHOT'
}

repositories {
    jcenter()
    maven { url "http://repo.spring.io/snapshot" }
    maven { url "http://repo.spring.io/milestone" }
}

dependencies {
    compile("org.springframework.boot:spring-boot-starter-web")
    testCompile("org.springframework.boot:spring-boot-starter-test")
}
```

9.2 Installing the Spring Boot CLI

The Spring Boot CLI is a command line tool that can be used if you want to quickly prototype with Spring. It allows you to run Groovy scripts, which means that you have a familiar Java-like syntax, without so much boilerplate code.

You don't need to use the CLI to work with Spring Boot but it's definitely the quickest way to get a Spring application off the ground.

9.2.1 Manual installation

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

- spring-boot-cli-1.2.0.M2-bin.zip
- spring-boot-cli-1.2.0.M2-bin.tar.gz

Cutting edge snapshot distributions are also available.

Once downloaded, follow the 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, or alternatively you can use `java -jar` with the `.jar` file (the script helps you to be sure that the classpath is set correctly).

9.2.2 Installation with GVM

GVM (the Groovy Environment Manager) can be used for managing multiple versions of various Groovy and Java binary packages, including Groovy itself and the Spring Boot CLI. Get `gvm` from gvmtool.net and install Spring Boot with

```
$ gvm install springboot
$ spring --version
Spring Boot v1.2.0.M2
```

If you are developing features for the CLI and want easy access to the version you just built, follow these extra instructions.

```
$ gvm install springboot dev /path/to/spring-boot/spring-boot-cli/target/spring-boot-cli-1.2.0.M2-bin/sp
$ gvm use springboot dev
```

```
$ spring --version
Spring CLI v1.2.0.M2
```

This will install a local instance of `spring` called the `dev` instance inside your gvm repository. It points at your target build location, so every time you rebuild Spring Boot, `spring` will be up-to-date.

You can see it by doing this:

```
$ gvm ls springboot

================================================================================
Available Springboot Versions
================================================================================
> + dev
* 1.2.0.M2

================================================================================
+ - local version
* - installed
> - currently in use
================================================================================
```

9.2.3 OSX Homebrew installation

If you are on a Mac and using Homebrew, all you need to do to install the Spring Boot CLI is:

```
$ brew tap pivotal/tap
$ brew install springboot
```

Homebrew will install `spring` to `/usr/local/bin`.

If you don't see the formula, your installation of brew might be out-of-date. Just execute `brew update` and try again.

9.2.4 Command-line completion

Spring Boot CLI ships with scripts that provide command completion for BASH and zsh 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. To run the script manually, e.g. if you have installed using `GVM`

```
$ . ~/.gvm/springboot/current/shell-completion/bash/spring
$ spring <HIT TAB HERE>
  grab  help  jar  run  test  version
```

If you install Spring Boot CLI using Homebrew, the command-line completion scripts are automatically registered with your shell.

9.2.5 Quick start Spring CLI example

Here's a really simple web application that you can use to test your installation. Create a file called `app.groovy`:

```
@RestController
class ThisWillActuallyRun {

    @RequestMapping("/")
    String home() {
        "Hello World!"
    }
```

```
}
```

Then simply run it from a shell:

```
$ spring run app.groovy
```

It will take some time when you first run the application as dependencies are downloaded. Subsequent runs will be much quicker.

Open localhost:8080 in your favorite web browser and you should see the following output:

```
Hello World!
```

9.3 Upgrading from an earlier version of Spring Boot

If you are upgrading from an earlier release of Spring Boot check the `release notes'' hosted on the project wiki. You'll find upgrade instructions along with a list of` new and noteworthy" features for each release.

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 remembering to update your `PATH` environment variable to remove any older references.

Prev	Up	Next
8. Introducing Spring Boot	Home	10. Developing your first Spring Boot application